ASP.NET MVC 5 開發實戰

開發環境與快速上手

多奇數位創意有限公司

技術總監 黃保翕 (Will 保哥)

部落格：http://blog.miniasp.com/

All services from your imperative.

ASP.NET MVC 5 開發環境建立

安裝必要軟體

All services from your imperative.

- Visual Studio 2015
  - Visual Studio 2015 Update 3
- Visual Studio 2013
  - Visual Studio 2013 Update 5
- Visual Studio 擴充套件
  - NuGet Package Manager (已內建)
  - Web Essentials 2013
  - VSCommands for Visual Studio 2013
  - tangible T4 Editor

Visual Studio 2013 範本選擇

All services from your imperative.

- .NET Framework 4
  - 傳統專案範本
    - ASP.NET Web Form 應用程式
    - ASP.NET MVC 4 Web 應用程式
- .NET Framework 4.5 / 4.5.1
  - 剩下一個：ASP.NET Web 應用程式
- 舊的專案範本
  - 整併到 Visual Studio 2012 分類下
  - .NET 4.5+：已移除 LINQ to SQL 專案範本！

Visual Studio 2015 範本選擇

All services from your imperative.

- .NET Framework 4
  - ASP.NET 空白 Web 應用程式
- .NET Framework 4.5+
  - ASP.NET Web Application
  - ASP.NET Core Web Application (.NET Core)
  - ASP.NET Core Web Application (.NET Framework)
- 舊的專案範本
  - 預設已經全數移除
  - 可搜尋相關範本專案

整合 ASP.NET 專案範本

All services from your imperative.

- 唯一 ASP.NET 專案範本
  - ASP.NET Web 應用程式 (僅限 .NET Framework 4.5+)
- 單一對話框
  - 選取範本
    - Empty , Web Forms , MVC , Web API , Single Page Application
  - 建立專案**可選取身分驗證與授權**的方式
    http://www.asp.net/visual-studio/overview/2013/creating-web-projects-in-visual-studio#auth
  - 透過**標準方式**即可自訂**專案範本**
    http://www.asp.net/mvc/overview/advanced/custom-mvc-templates

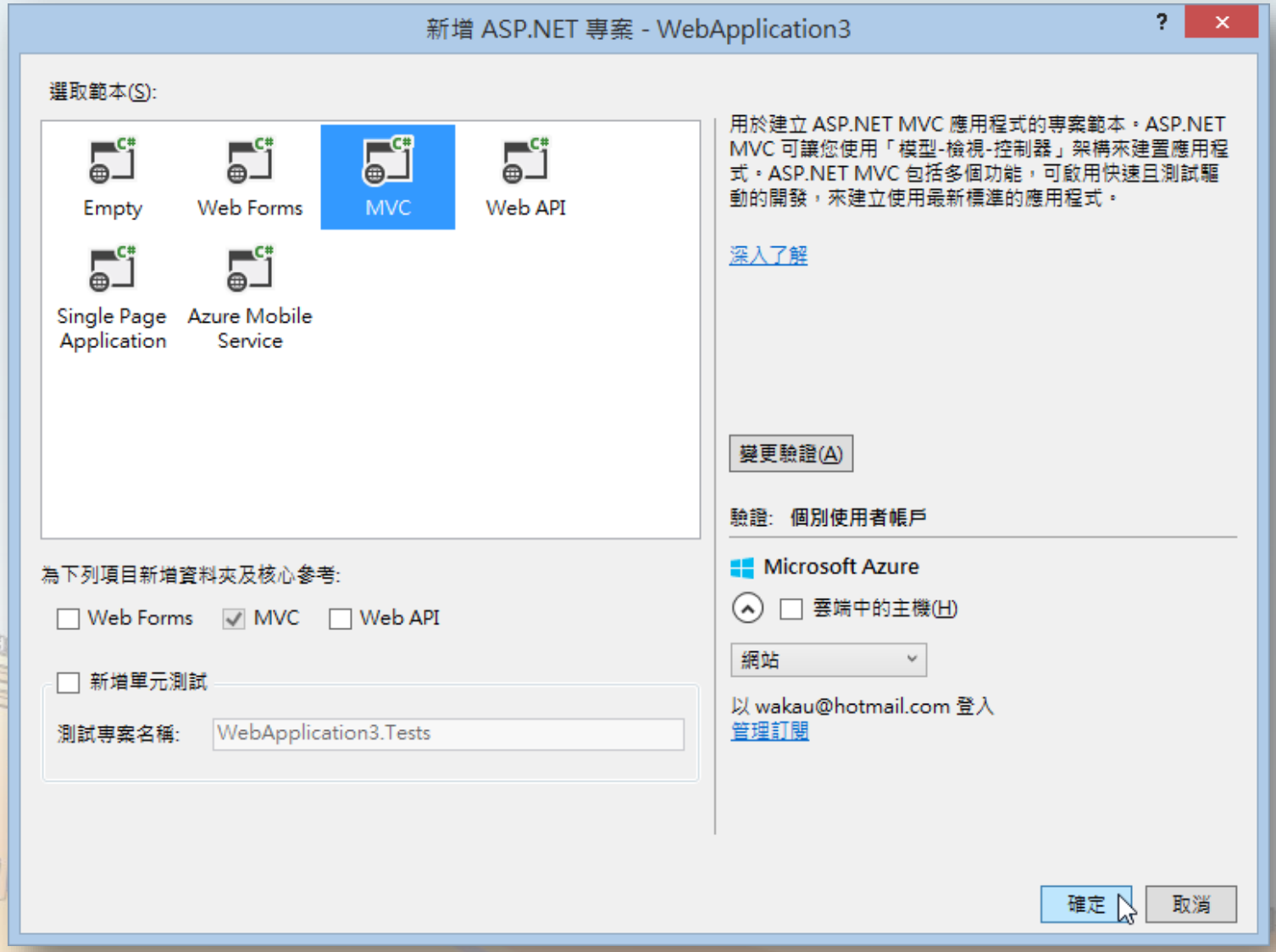
介紹 ASP.NET MVC 5 專案範本

All services from your imperative.

- 專案範本 (Project Template)
  - Empty
  - Web Forms
  - MVC
  - Web API
  - Single Page Application
- 內建多種常用的套件 (packages.config)
  - 包括 Bootstrap、jQuery、 jQuery.Validation、 Modernizr、 Newtonsoft.Json、Entity Framework 、…
  - 套件皆可搭配 NuGet 自動升級!

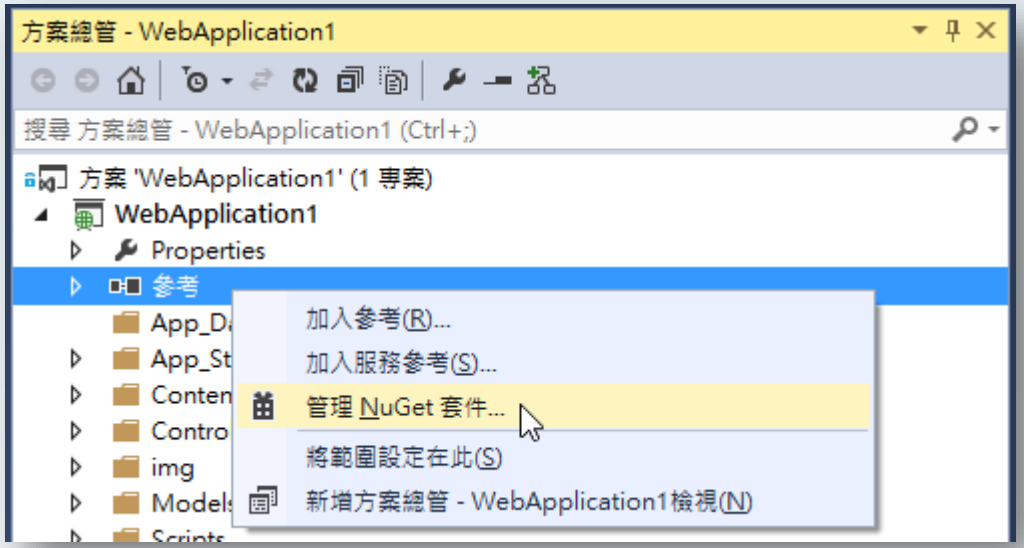
介紹 ASP.NET MVC 5 專案範本

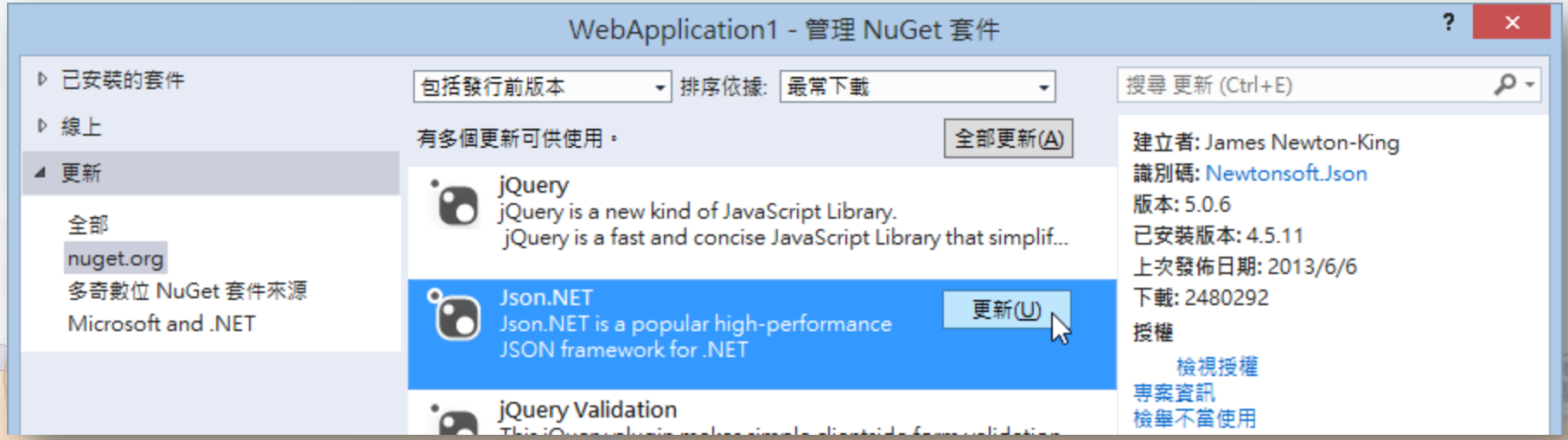
All services from your imperative.



透過 NuGet 升級內建套件

All services from your imperative.

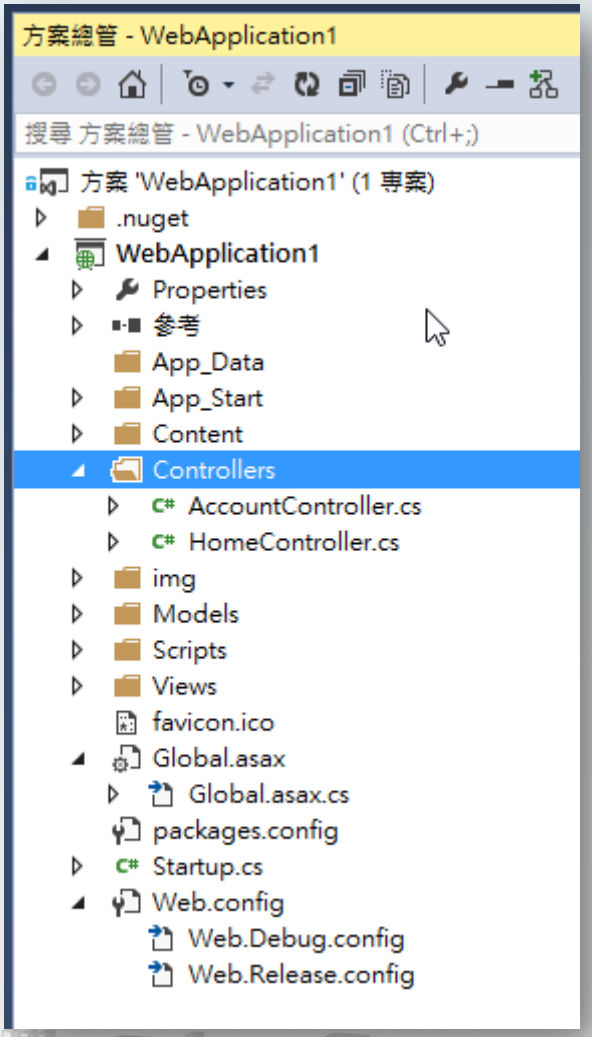




專案範本結構介紹

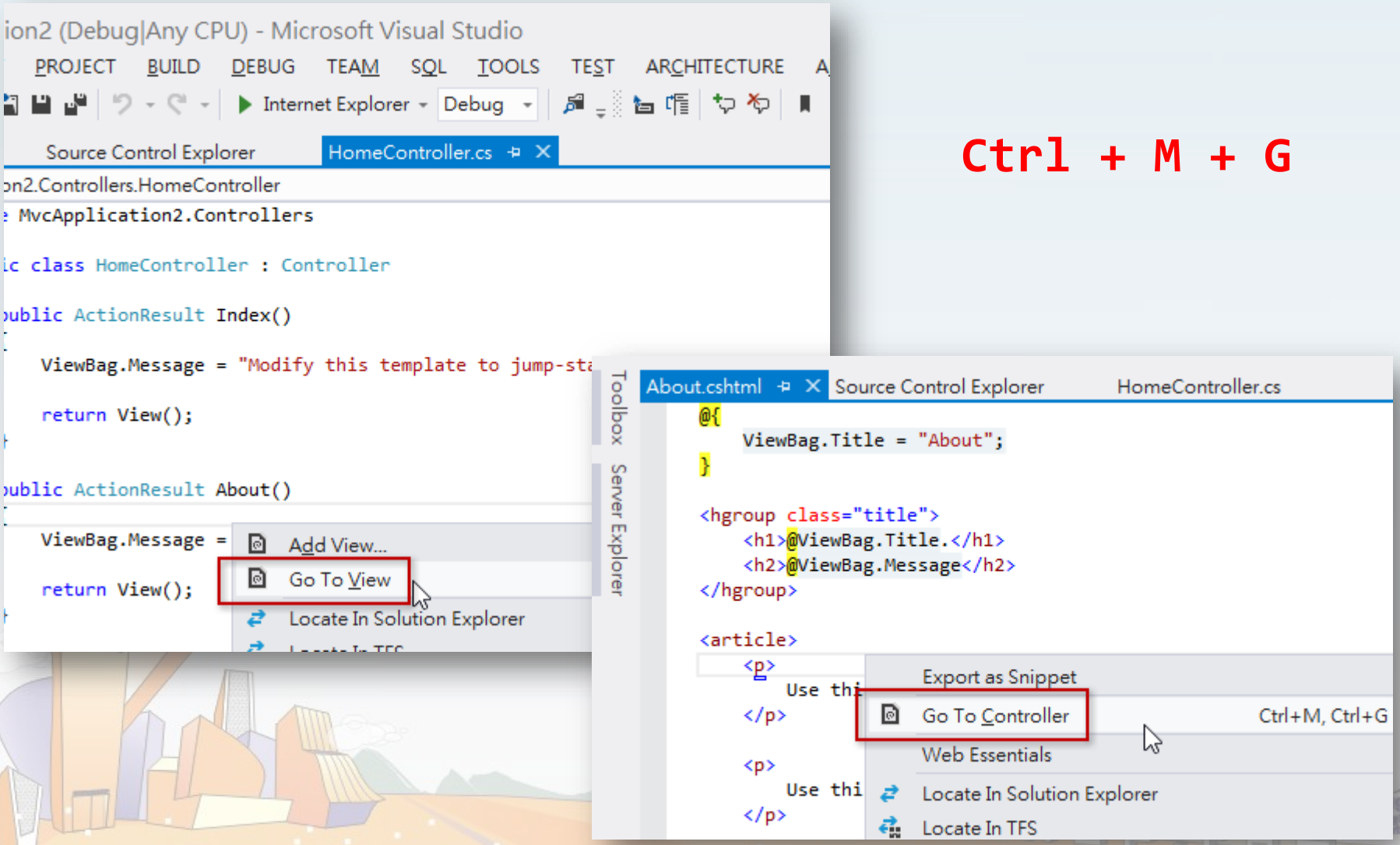
All services from your imperative.

- 主要目錄
  - Controllers
  - Models
  - Views
  - **App_Start**
- 資源檔案目錄
  - App_Data
  - Content
  - Scripts
- 重要檔案
  - Web.config
  - Global.asax.cs
  - packages.config
  - Startup.cs



快速於 Controller 與 View 之間切換

All services from your imperative.



Ctrl + M + G

All services from your imperative.

實戰演練 ASP.NET MVC 開發

ASP.NET MVC 5 開發實戰

ASP.NET MVC 架構與觀念解析

多奇數位創意有限公司

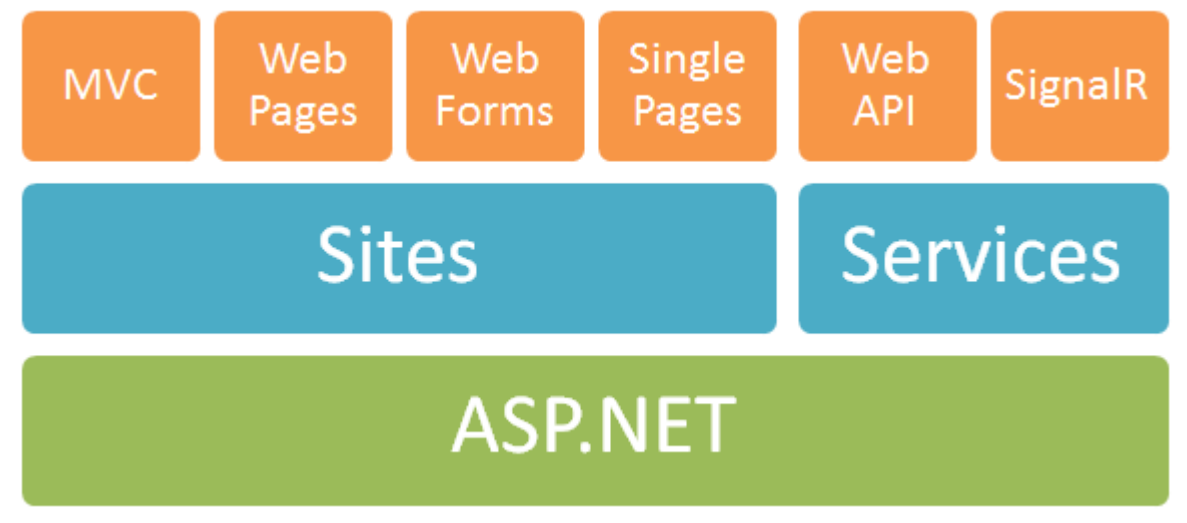
技術總監 黃保翕 (Will 保哥)

部落格：http://blog.miniasp.com/

瞭解 ASP.NET 架構與發展狀況

All services from your imperative.

ASP.NET 發展現況



All services from your imperative.

了解 WEB FORM 與 MVC 的差異

Web Form 與 ASP.NET MVC 的差異

All services from your imperative.

支援功能	ASP.NET Web Form	ASP.NET MVC
ViewState	☑	☐
ASP.NET 頁面追蹤機制 (Page Trace)	☑	☐
ASP.NET 事件驅動模型 (Event Model)	☑	☐
伺服器控制項 (Server Control)	☑	部分支援
System.Web.SiteMapProvider 類別	☑	☐
ASP.NET Provider Model	☑	☑
System.Web.Caching 命名空間	☑	☑
System.Web.SessionState 命名空間	☑	☑
System.Web.Security 命名空間	☑	☑
Profile, Membership, SiteMap	☑	☑
其他 System.Web.* 功能	☑	☑

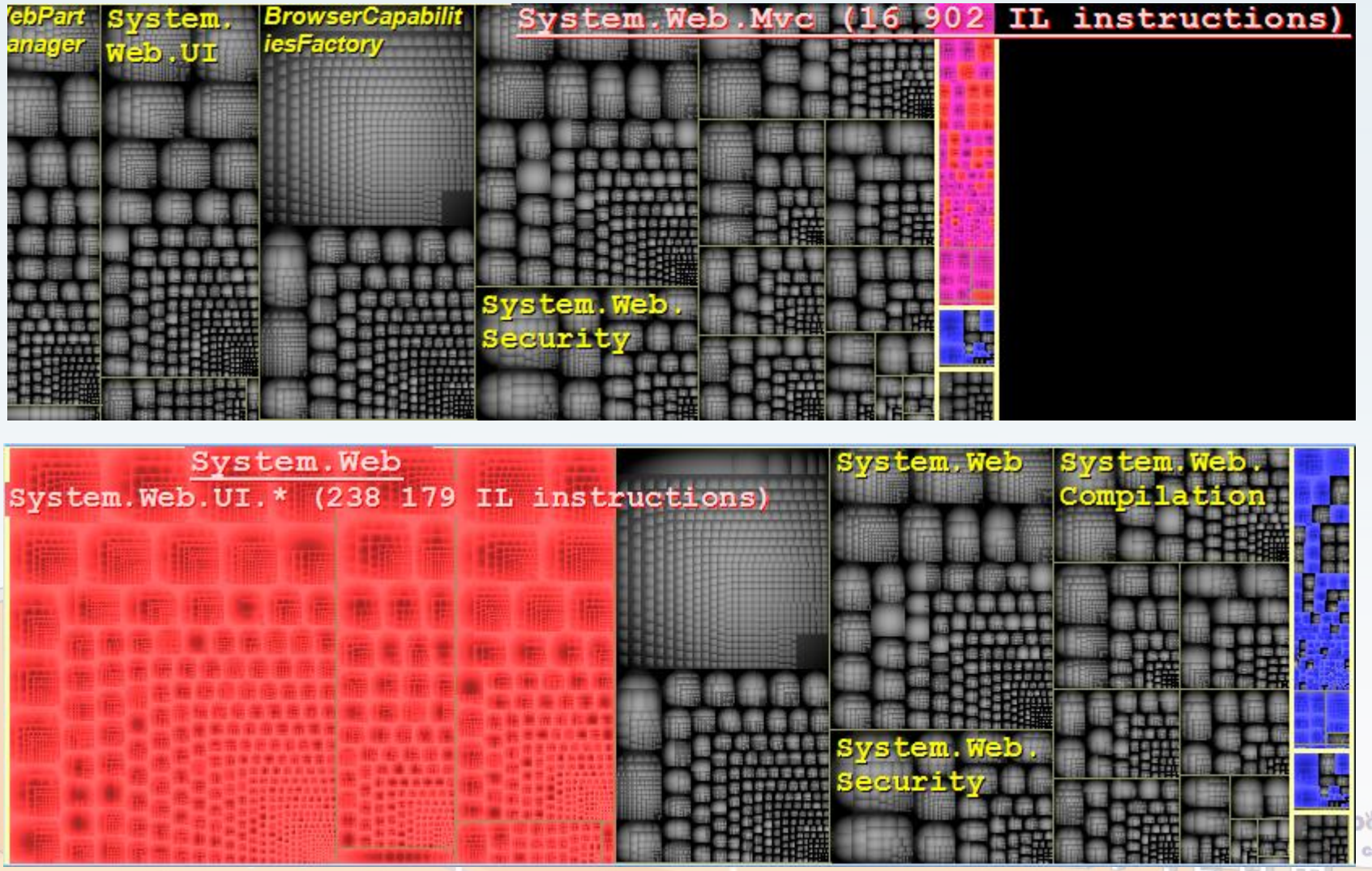
Web Form 與 ASP.NET MVC 的差異

All services from your imperative.

- 相同之處
  - 都是 IHttpHandler 的產物
  - 共享 ASP.NET Framework 的所有功能
- IL 指令集差很多
  - System.Web.Mvc.*: 16,902 IL instructions
    - 雖然 ASP.NET MVC 的 WebFormView 用到許多原本 WebForm 的東西，但是還是很多控制項不會去用他！
    - ASP.NET MVC 的 ViewPage 繼承 System.Web.UI.Page (非常邪惡)
  - System.Web.UI.*: 238,179 IL instructions

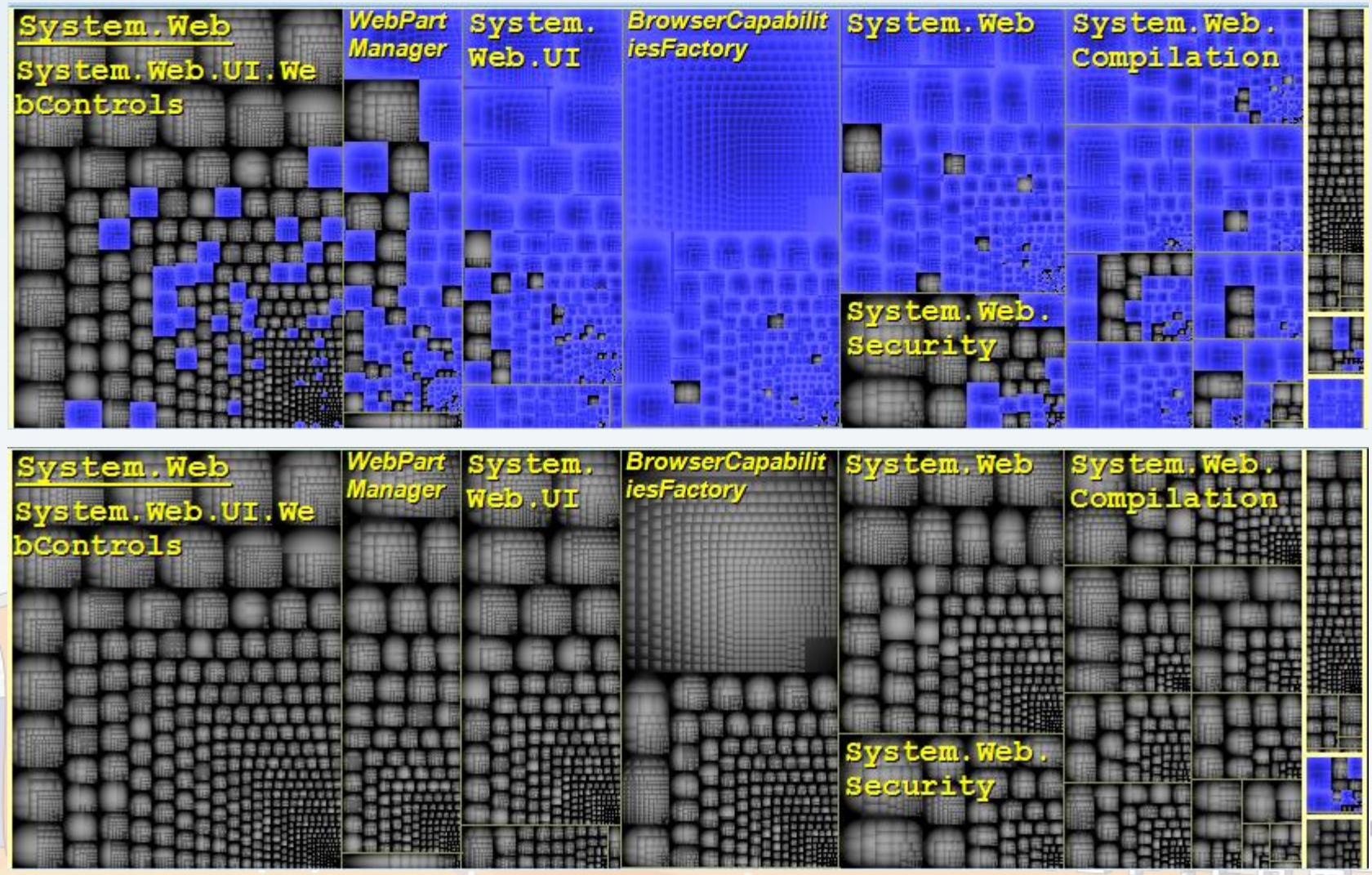
指令集數量的差別

All services from your imperative.



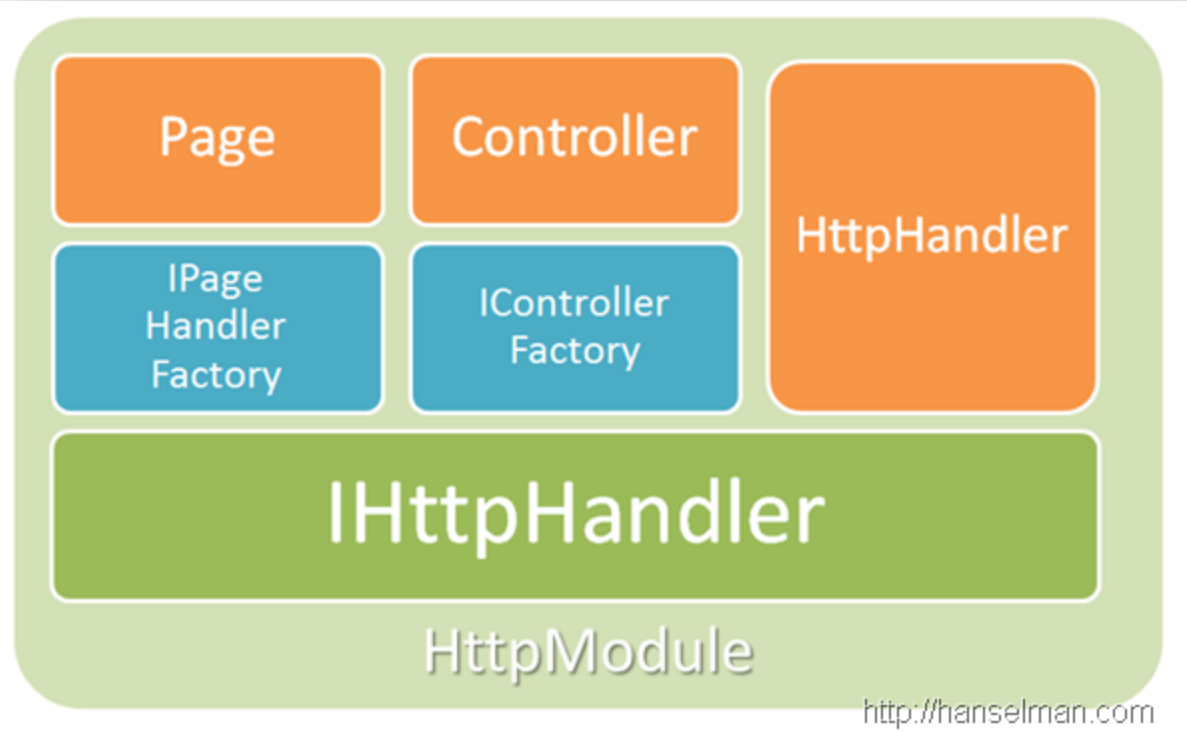
有沒有 WebFormView 的差別

All services from your imperative.



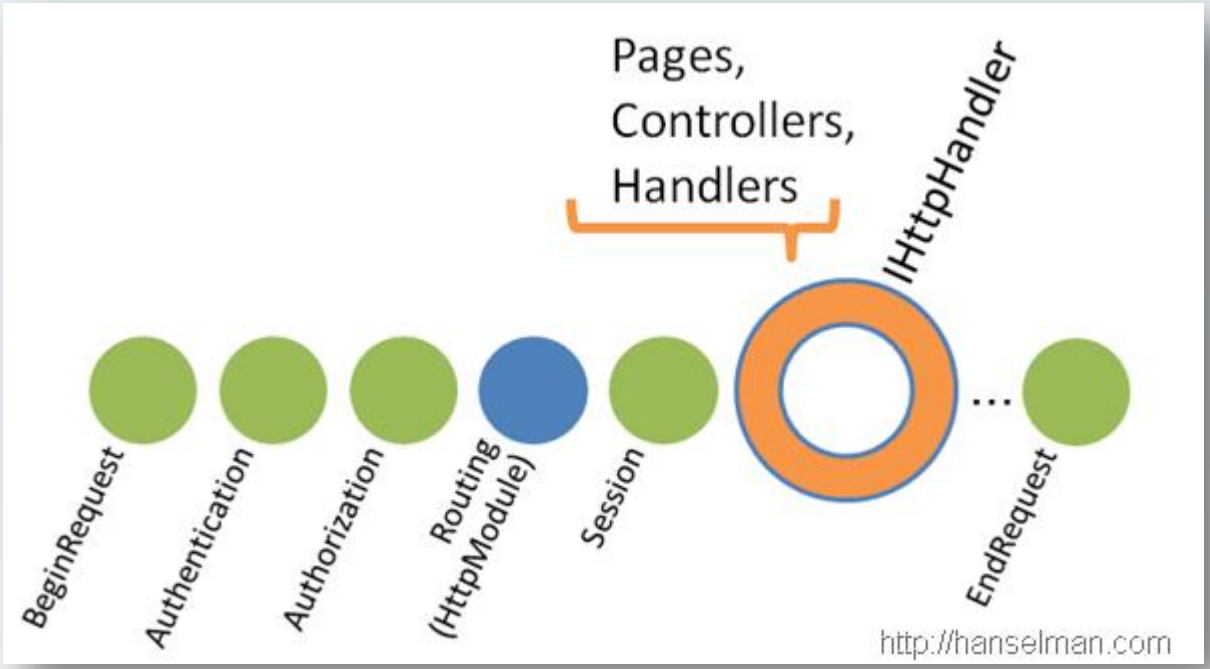
HttpModule 與 HttpHandler

All services from your imperative.



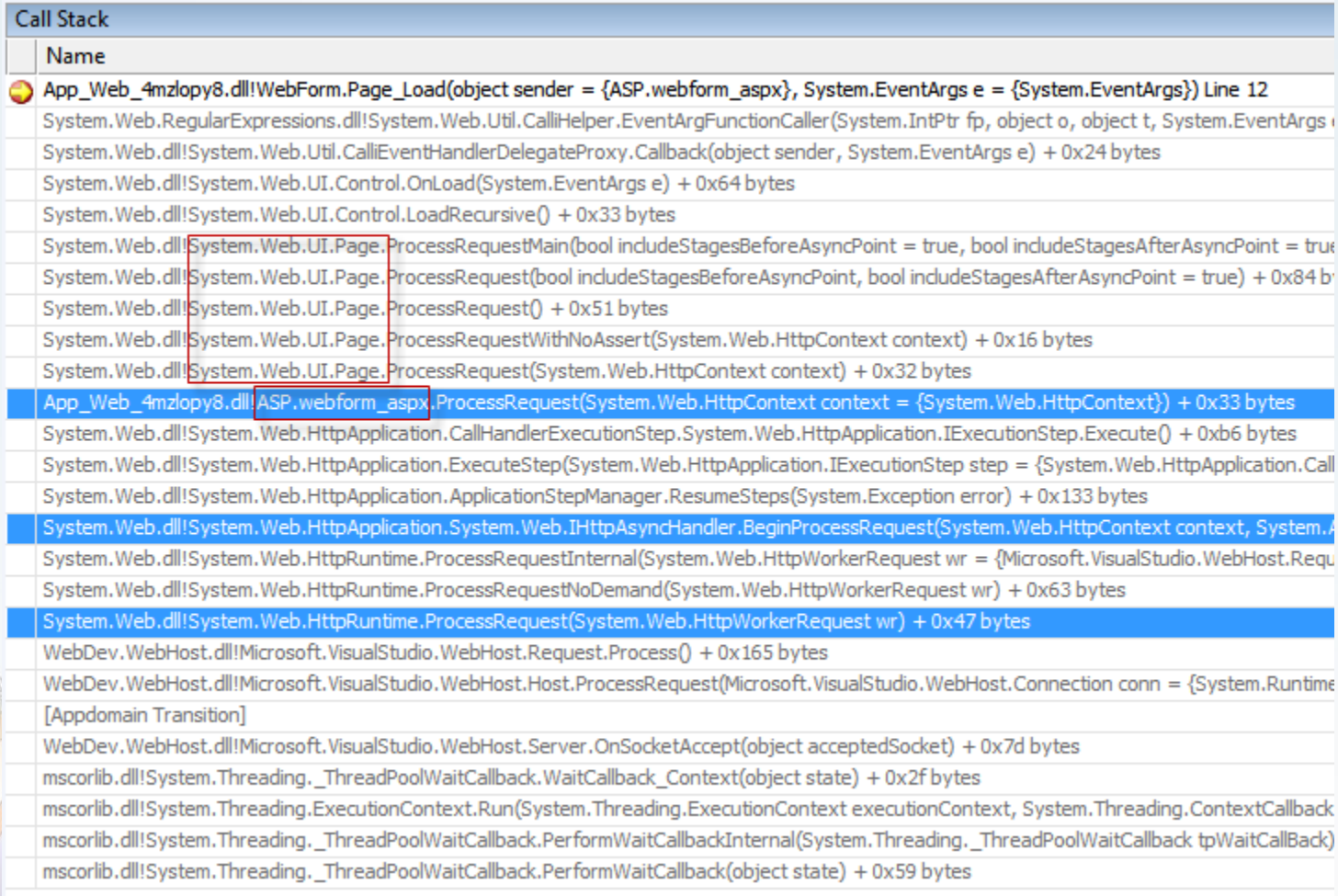
ASP.NET 執行的過程

All services from your imperative.



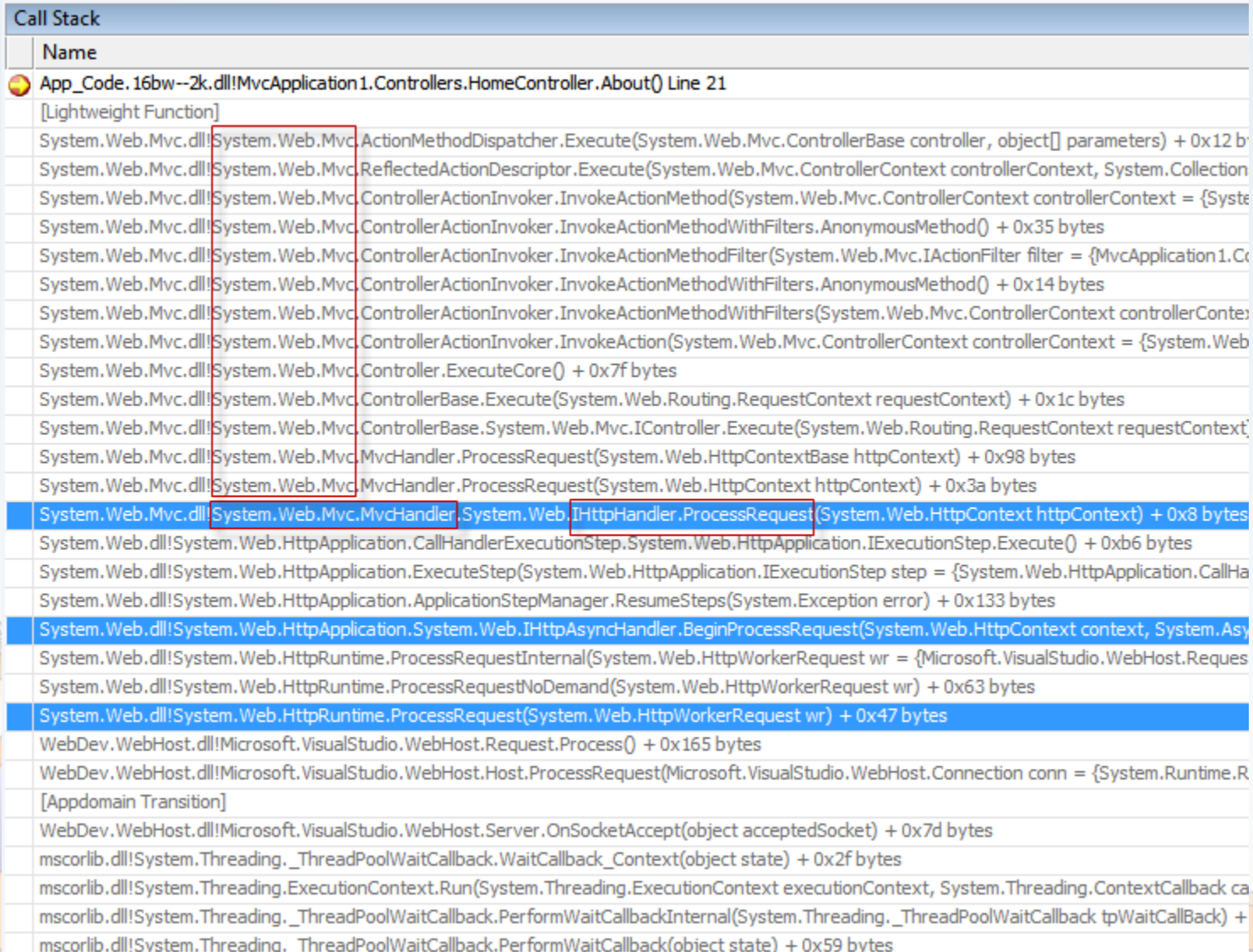
WebForm 的執行過程

All services from your imperative.



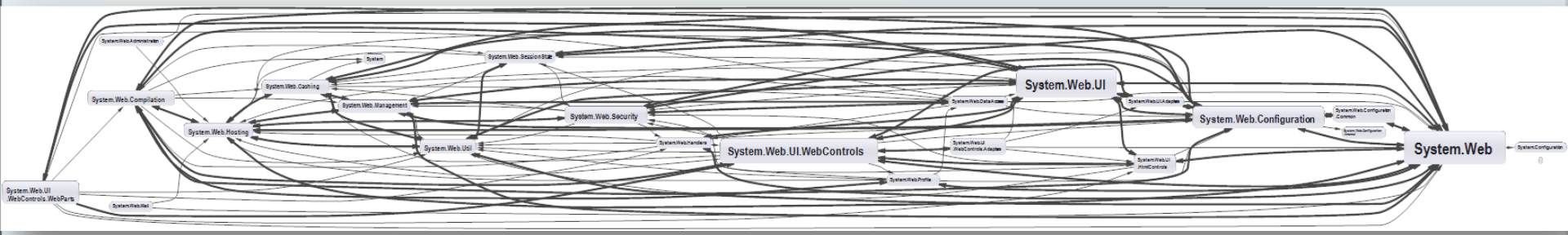
ASP.NET MVC 的執行過程

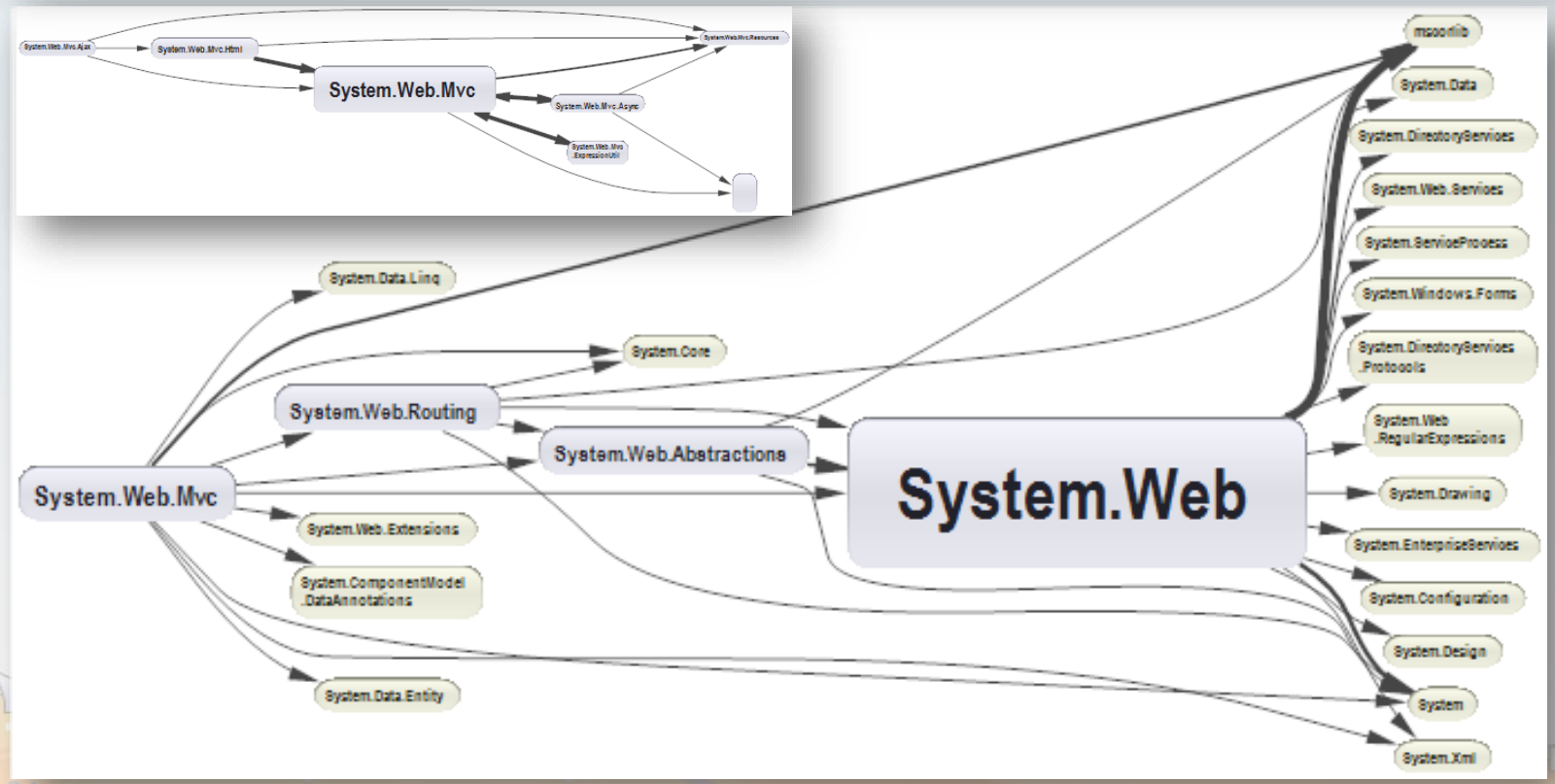
All services from your imperative.



Web Form 與 ASP.NET MVC 的差異

All services from your imperative.



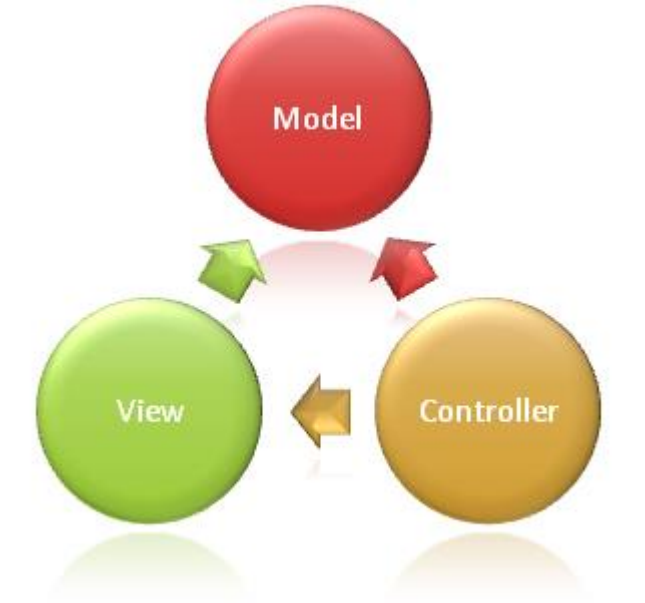


All services from your imperative.

ASP.NET MVC 核心觀念

瞭解 MVC 設計樣式

All services from your imperative.



關注點分離 (Separate of Concerns)

All services from your imperative.

- 簡化複雜度
  - 將複雜的問題拆解成數個容易解決的單元
  - 一次僅關注於一個較容易理解與解決的部分
- 可維護性大幅提昇
  - 不止可區分 Model、View、Controller
  - 還可以再切割成更多層次，讓專案更易於維護
- 更容易測試
  - 僅測試需要關注的點
  - 測試導向先行 (Test-Driven Development; TDD)

以習慣取代配置

All services from your imperative.

- 約定俗成的開發習慣
  - 能將繁瑣的開發流程簡化
  - 有效減少溝通與決策的時間，降低溝通成本

- 培養團隊的習慣
  - 瞭解 ASP.NET MVC 的開發習慣極其重要
  - 開發團隊能有自己的開發習慣
    - 方案規劃、命名空間、命名規則、目錄結構
    - 使用 NuGet、Entity Framework、EFRepository、MVC Turbine、…
    - 不要在 View 裡使用 LINQ to SQL 的「點表示法」(dot notation)
    - 適度的使用 ViewModel

控制反轉 (Inversion of Control)

All services from your imperative.

- 專注於軟體設計
  - 讓各模組之間透過 合約 (contracts) 進行合作
  - 各模組無法對其他模組進行假設，因而專注於實作
  - 組合模組的人可專注於介面設計，而非實作細節
  - 替換任何模組時不會有副作用 (side effect)
- 特性
  - 執行時期 (runtime) 才會注入 (injects) 實體
  - 透過相依性注入 (Dependency injection) 實現 IoC
  - 較容易落實「關注點分離」但也容易變複雜

ASP.NET MVC 擴充點 - Model

All services from your imperative.

- 擴充點 (extensibility points)
  - Model
    - ValueProvider　　輸入數值轉換成資料
    - ModelBinder　　輸入資料轉換成型別
      - DefaultModelBinder 不支援 immutable objects
    - ModelMetadata
    - DataAnnotation / IValidatableObject
      - ValidationAttribute
        - DataAnnotationsModelValidatorProvider
      - IDataErrorInfo
        - DataErrorInfoModelValidatorProvider
      - ClientDataTypeModelValidatorProvider
      - ModelValidatorProvider

ASP.NET MVC 擴充點 - View

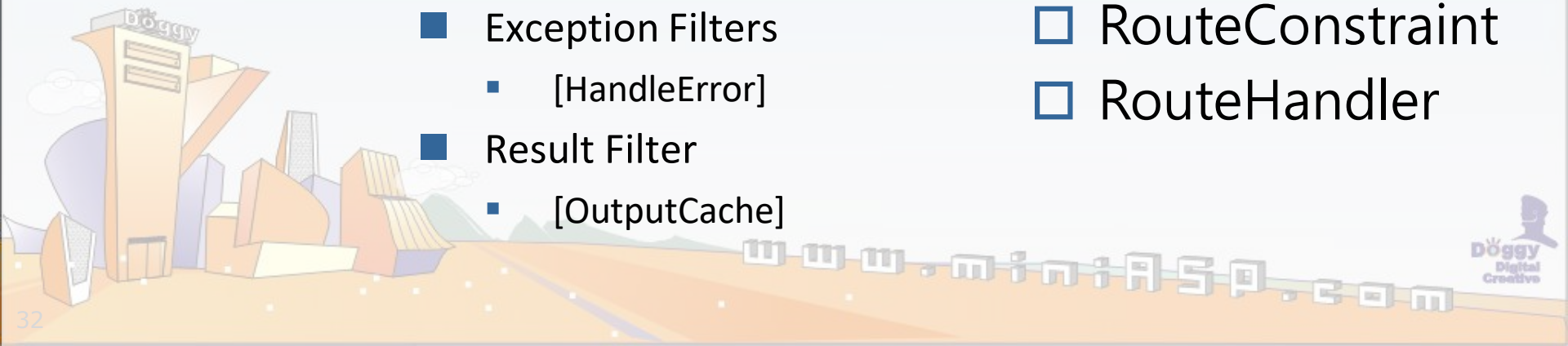
All services from your imperative.

- 擴充點 (extensibility points)
  - View
    - ViewEngine : VirtualPathProviderViewEngine
      - WebFormViewEngine
      - RazorViewEngine
    - HTML Helper
    - Razor Helper
  - Controller

ASP.NET MVC 擴充點 - Controller

All services from your imperative.

- 擴充點 (extensibility points)
  - Controller
    - Action Selectors
      - [ActionName]
      - [ActionMethodSelector]
    - Action Filters
      - Authorization Filters
        - [Authorize]
      - Exception Filters
        - [HandleError]
      - Result Filter
        - [OutputCache]
    - ControllerFactory
    - ControllerBase
    - Action Results
  - Routing
    - RouteConstraint
    - RouteHandler



分層開發

All services from your imperative.

- 透過 MVC 分階段開發
  - Model → Controller → View
  - Model → Repository → Controller → View
  - Model → Repository → Service → Controller → View
  - IoC / DI / AutoMapper / Unit Test / Integration Test
- 透過**關注點分離**簡化複雜度
  - 漸進式擴充各 ASP.NET MVC 擴充點
  - HTML Helper, AtionFilter, Model Binder, Model Validation, …
- 定時做 Code Refactoring
  - 推薦書籍：重構 - 向範式前進 (Refactoring to Patterns)

團體合作的ASP.NET MVC開發策略

All services from your imperative.

- 架構師專注 Model
  - 設計資料結構、資料驗證規則、商業邏輯
  - LINQ to SQL, Repository, Service, DAL
  - Model Validation, Model Binder, DataAnnotation
- 分析師專注 Controller
  - 設計網頁與後端的互動關係
  - URL Routing, Controller, Actions, Action Filter
  - 撰寫單元測試程式 (非整合測試)
- 開發人員專注 View
  - 決定前端呈現的細節、實做Action的程式碼
  - 開發人員千萬不要自己承攬 HTML, CSS 的編修工作

All services from your imperative.

ASP.NET MVC 執行生命週期

ASP.NET MVC 執行生命週期(簡化版)

All services from your imperative.

HTTP 要求	• http://example.com/Product/Detail/1 • GET /Product/Detail/1
Controller	• ProductController.Detail(1)
Model	• ProductRepository.GetProduct(1)
View	• ~/Views/Product/Detail.aspx
HTTP 回應	• HTML, XML, RSS, Text, CSS, File (Binary), …

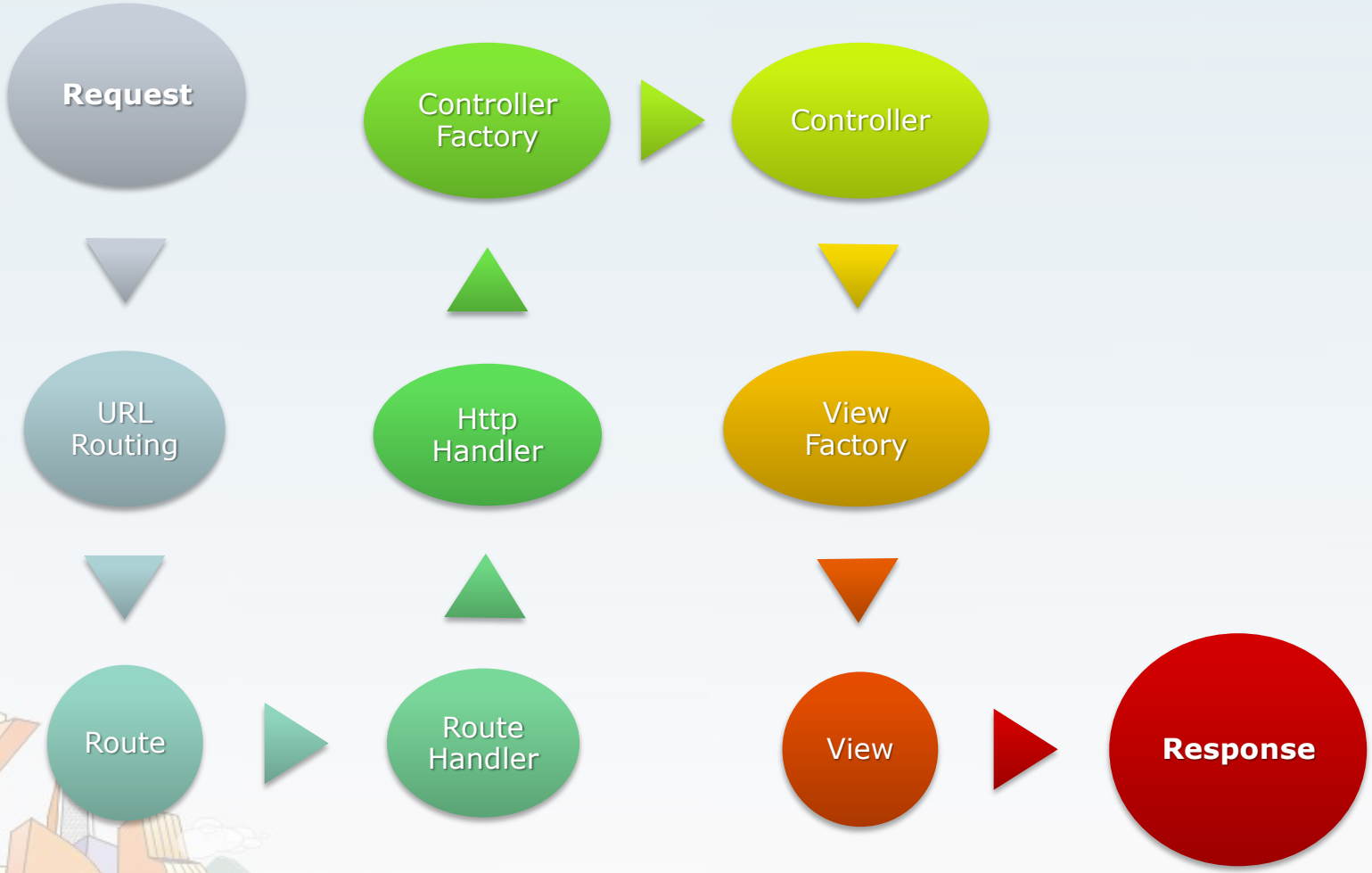
ASP.NET MVC 執行生命週期(簡化版)

All services from your imperative.

HTTP 要求	• POST /Product/Edit/1
Controller	• ProductController.Edit(1, FormCollection c) • TryUpdateModel(product, new string[] { "Name"})
Model	• var product = ProductRepository.GetProduct(1) • ProductRepository.UpdateProduct(1, product)
View	• N/A
HTTP 回應	• HTTP 重導向(Redirection)

瞭解 ASP.NET MVC 執行生命週期

All services from your imperative.



ASP.NET MVC 5 開發實戰

資料模型開發實務

多奇數位創意有限公司

技術總監 黃保翕 (Will 保哥)

部落格：http://blog.miniasp.com/

Model

瞭解資料模型、ORM 與 POCO

資料模型 (Model)

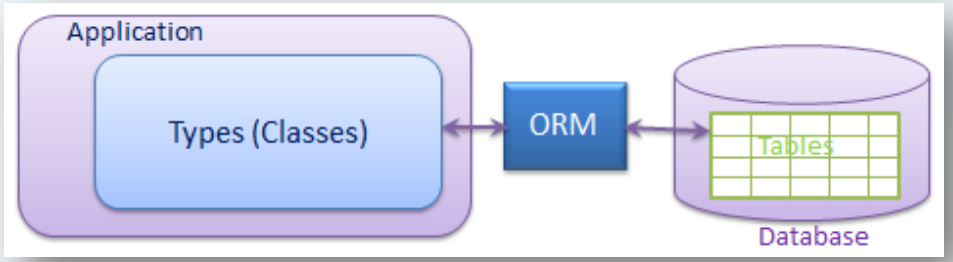
All services from your imperative.

- 描述資料的一種形式 (**模型**是一種**抽象概念**)
  - XML
    - DOCTYPE
  - Class
    - Fields, Properties
  - Table Schema
    - Columns
  - Custom Format
- ViewModel
  - 針對 View 所需而設計的 Model

ORM (Object-relational mapping)

All services from your imperative.

- ORM 做什麼事？
  - 將**結構化的關連資料**對映成**物件導向模型**
  - 將**物件資料**對應成**關連資料**
- 市面上有哪些 ORM 技術？
  - Typed DataSet
  - LINQ to SQL
  - Entity Framework
  - Devart LinqConnect
  - NHibernate
  - SubSonic



… and much more …

採用 ORM 有什麼好處

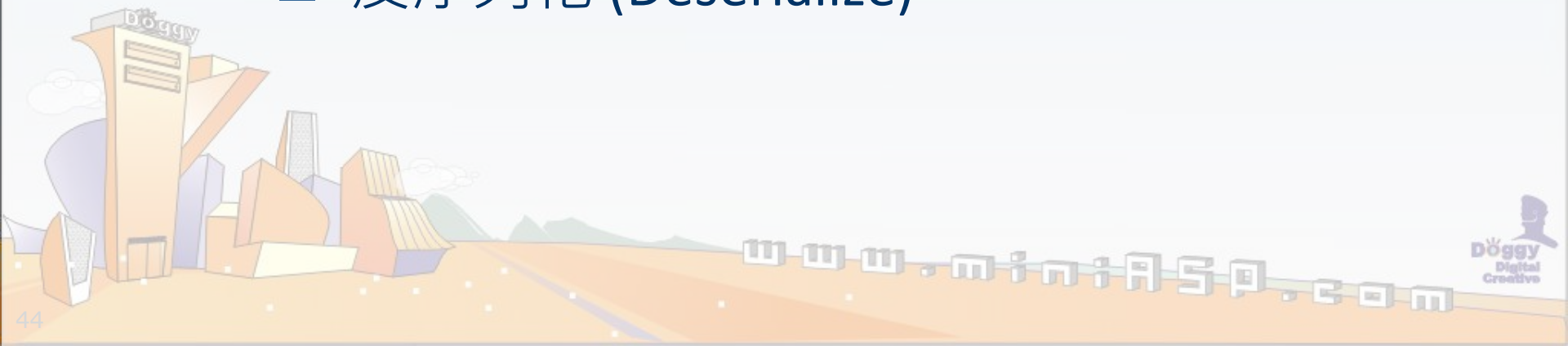
All services from your imperative.

- 對 DBA 來說
  - 更專注於自身的 DBA 生涯磨練
- 對開發人員來說
  - 更專注於增強開發技巧
  - 不再侷限於現有資料庫架構
    - 以前: 對開發人員來說 DBA 越強勢越痛苦 (太沒結構)
    - 現在: 透過 ORM 技術開發不再受侷限
  - 可以再開發時期快樂的且安全的重構
- 想達到的目的：專業分工
  - 開發人員可以不懂 T-SQL / Stored Procedure
  - DBA 可以不懂 OOP、.NET、**Entity Framework**

POCO (Plain Old CLR Object)

All services from your imperative.

- 在 ORM 領域中經常被使用
- 用最簡單且無負擔的方式表達資料模型
- 可當 DTO (Data Transfer Objects) 使用
  - 序列化 (Serialize)
  - 反序列化 (Deserialize)

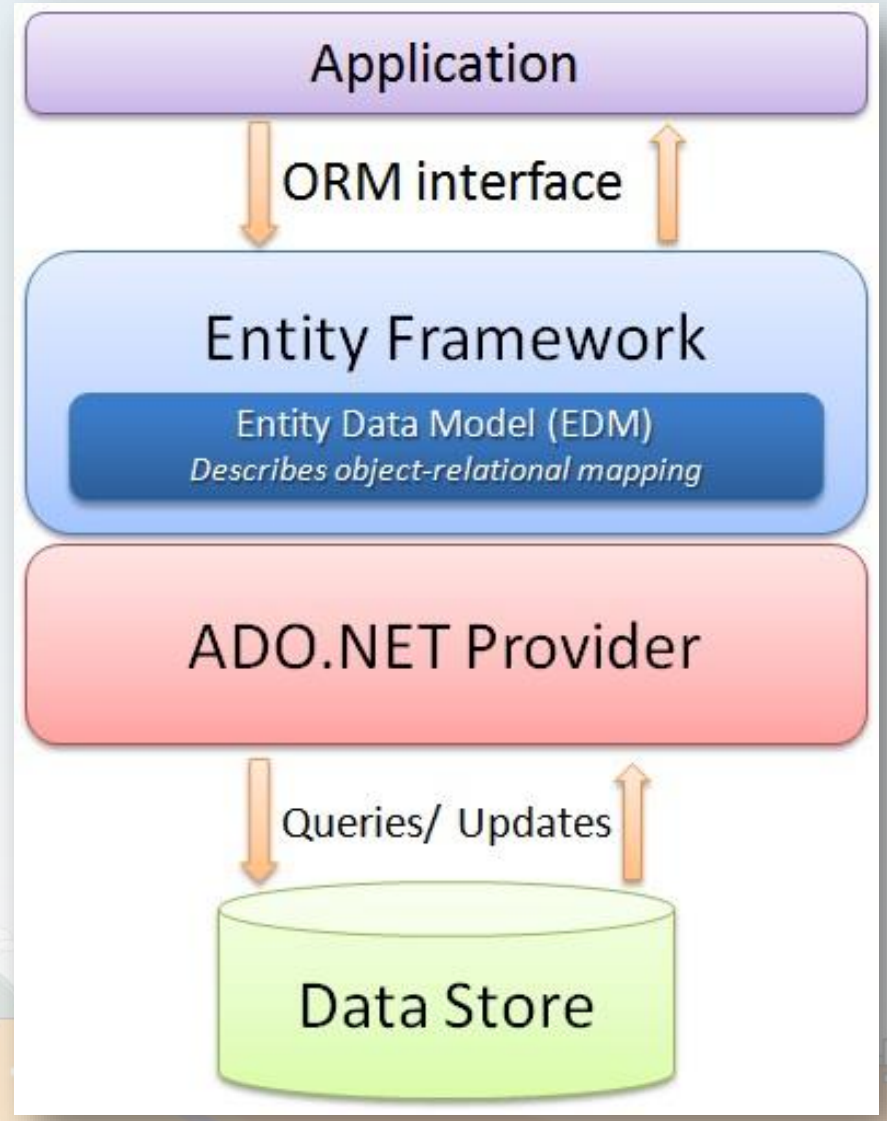


Introducing to Entity Framework

ENTITY FRAMEWORK 概論

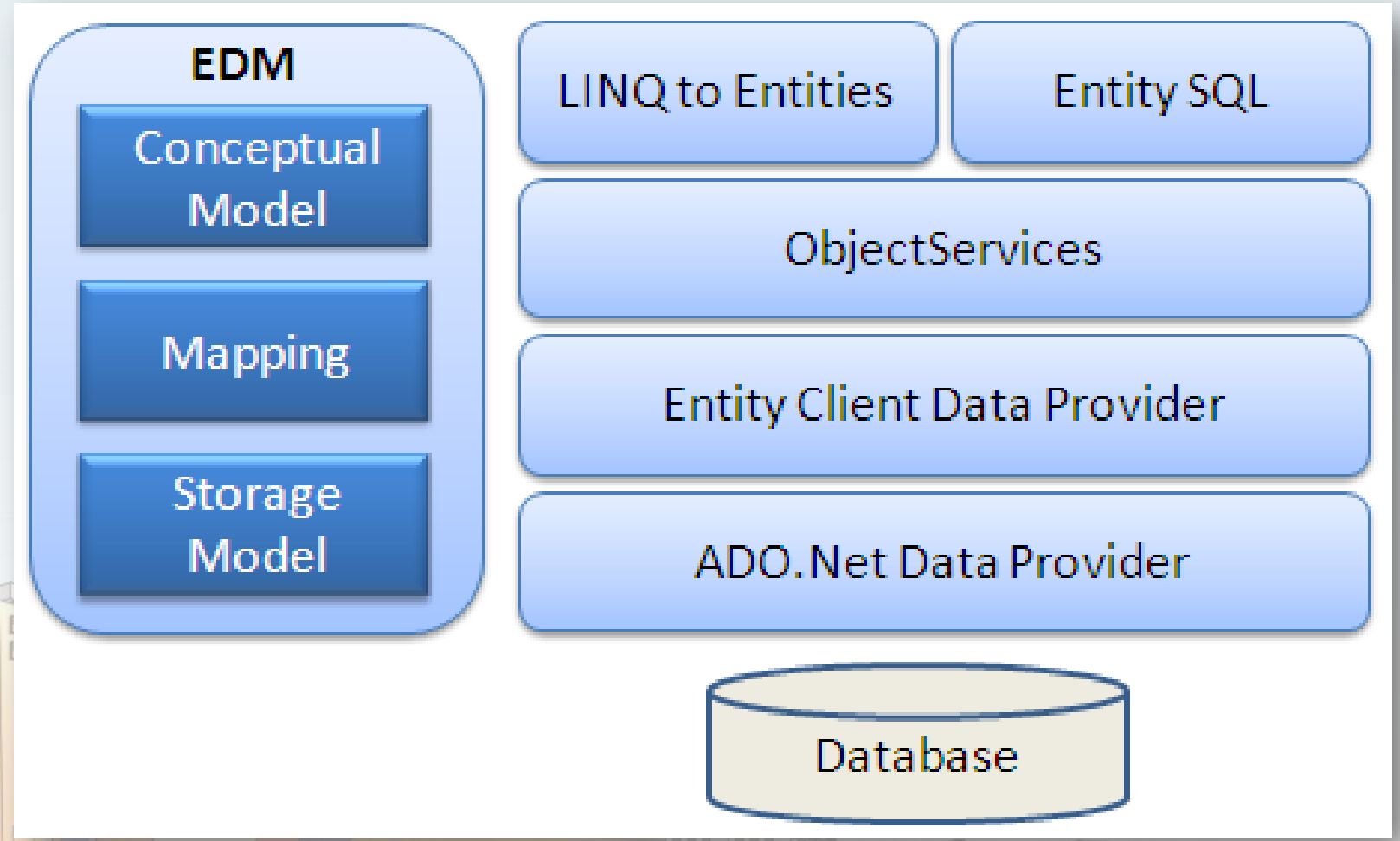
Entity Framework 架構

All services from your imperative.



Entity Framework 框架

All services from your imperative.



Entity Framework 程式碼片段

All services from your imperative.

- 新增資料
  - `Client c = new Client();`
  - `c.fname = "Will";`
  - `c.lname = "Huang";`
  - `c.phone = "2222-3333";`

  - `FabricsEntities db = new FabricsEntities();`
  - `db.Client.Add(c);`
  - `db.SaveChanges();`

Entity Framework 程式碼片段

All services from your imperative.

讀取資料

```
FabricsEntities db = new FabricsEntities();
var all = db.Client.AsQueryable();
var one = all.Find(1); // P.K. Only
var some = all.Where(p=>p.Id == 1);
var first = all.First(Lambda);
var first = all.FirstOrDefault(Lambda);
var single = all.Single(Lambda);
var single = all.SingleOrDefault(Lambda);
var take5 = all.Take(5);
var orderby = all.OrderBy(p => p.Id);
var page2 = orderby.Skip(10).Take(10);
```

Entity Framework 程式碼片段

All services from your imperative.

- 更新資料
  - `FabricsEntities db = new FabricsEntities();`
  - `var c = db.Client.Find(1);`
  - `c.City = "Taipei";`
  - `db.SaveChanges();`

Entity Framework 程式碼片段

All services from your imperative.

刪除資料

- `FabricsEntities db = new FabricsEntities();`
- `var c = db.Client.Find(1);`
- `db.Client.Remove(c);`
- `db.SaveChanges();`

練習 Entity Framework 資料操作

All services from your imperative.

- 新增一筆 Client 資料
- 將所有 Client 資料的 City 欄位改成**台北市**
- 刪除前 5 資料
  - 注意 SQL 的關聯限制 (Foreign Key Constrain)
  - 透過 RemoveRange 刪除關聯資料

Entity Framework 的開發方式

All services from your imperative.

- Entity Framework 支援三種開發導向
  - 資料庫先行開發模式(**Database First** Development)
    - DBA is the king!
  - 模型先行開發模式 (**Model First** Development)
    - DBA is helping.
  - 程式碼先行開發模式 (**Code First** Development)
    - DBA is nothing……

Database First Development

All services from your imperative.

- 如何設計「資料庫」才能幫助開發人員？
  - 一定要有 Primary Key
  - 資料型別要定義完整 (想清楚)
    - 欄位型態與最大長度
    - 允許 NULL
    - 預設值
  - 表格之間要建立適當的關連 (Foreign Keys)
  - 依據作業需求設計合適的檢視表 (Views)
  - 為**複雜的查詢**或**資料操作**撰寫**預儲程序**
  - 有一致的命名規則 (可用中文命名)

Database First Development

All services from your imperative.

- 執行「**從資料庫更新模型**」為何不更新？
  - 新增表格
    - 會自動新增相關定義到 SSDL、CSDL 與 MSL 定義中
  - 刪除表格
    - 會自動刪除 SSDL 與 MSL 中所以與該表格相關的定義，CSDL 不會更新
  - 更新表格 (新增欄位)
    - 會自動新增相關定義到 SSDL、CSDL 與 MSL 區段中
  - 更新表格 (刪除欄位)
    - 會自動自動刪除 SSDL 與 MSL 中所有該欄位的定義，CSDL 不會更新
  - 更新表格 (欄位更名)
    - 相當於先 **刪除欄位** 再 **新增欄位**
      - 舊欄位：會自動自動刪除 SSDL 與 MSL 中所有該欄位的定義，CSDL 不會更新 (需手動更新)
      - 新欄位：會自動新增相關定義到 SSDL、CSDL 與 MSL 區段中
  - 更新表格 (欄位變更屬性，例如: NOT NULL –> NULL)
    - 會自動自動更新 SSDL 中所有該欄位的定義，CSDL 也會更新
      (註: 這種情況下 MSL 本來就不需要更新)

Model First Development

All services from your imperative.

- 如何設計「模型」？
  - 一定要有 Entity Key (主鍵值)
  - 資料型別要定義完整 (想清楚)
    - 欄位型別 (Type) 與 最大長度 (Max Length)
    - 允許 NULL (Nullable)
    - 預設值 (DefaultValue)
      註: 並非資料庫的**預設值**設定!
  - 模型之間要建立適當的關連 (Association)
  - 有一致的命名規則 (可用中文命名)
- 儲存後會自動產生 *.cs 類別檔 (POCO)
- 最後才產生**各資料庫**所需的 DDL 指令碼

Code First (Code Only) Development

All services from your imperative.

- 如何設計「類別」？
  - **完全**從物件的角度思考
  - 設計的方法
    - 決定屬性的型別與名稱
    - 套用欄位驗證屬性 (Attributes)
    - 類別繼承 (inheritance)
    - 類別屬性相依 (複雜型別) (Complex Type)
    - 套用延遲載入特性 (virtual)
    - 最後設計 DbContext 產生資料集合
- 資料庫全自動產生，只需調整**連線參數**

導入Entity Framework 後 DBA 能做的事

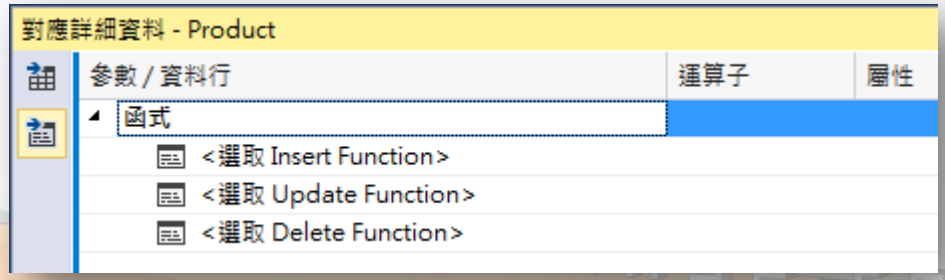
All services from your imperative.

- 原本的資料庫維運工作不能少
- 協助開發人員對資料庫進行分析與效能調校
  - 索引與查詢效能
  - 分析交易與鎖定
  - T-SQL 語法調校？
  - **執行計畫與查詢計畫？**
    - Entity Framework 學習心得分享 (以 LINQ to SQL 為基礎)
      http://blog.miniasp.com/post/2009/03/01/Entity-Framework-Learning-Notes-based-on-LINQ-to-SQL.aspx

Entity Framework 效能調校

All services from your imperative.

- 非常有限度的調整 T-SQL 語法
  - 查詢計畫
    - `db.News.Include("Category")`
    - `using System.Data.Entity;`
      `db.News.Include(s => s.Category)`
  - 執行原始查詢 (無回傳型別) (DDL/DML)
    - `db.Database.ExecuteSqlCommand(strSQL, params…)`
  - 建立原始查詢 (帶有回傳型別) (SELECT/SP)
    - `db.Database.SqlQuery<T>(strSQL, params…)`
- 善用預存程序優化查詢效率
  - Insert
  - Update
  - Delete



Input Validation & Model Validation

輸入驗證 v.s. 模型驗證

擴充資料模型

All services from your imperative.

- 開發步驟
  - 建立 Client.Partial.cs
  - 內容來自 Client.Partial.cs
  - 類別名稱加上 Metadata
  - 建立 Partial 類別
並套用 MetadataType 屬性
  - 開始套用**驗證屬性**

會員資料.Partials.cs* 會員資料.cs

MvcApplication1.Models.會員資料Metadata

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace MvcApplication1.Models
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;

    [MetadataType(typeof(會員資料Metadata))]
    public partial class 會員資料
    {
    }

    public class 會員資料Metadata
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public System.DateTime Birthday { get; set; }
    }
}
```

使用 DataAnnotations 定義欄位驗證

All services from your imperative.

- 引用命名空間
  - System.ComponentModel.DataAnnotations
- 組件名稱
  - System.ComponentModel.DataAnnotations.dll
- 欄位驗證屬性
  - [MinLength]
  - [MaxLength]
  - [CreditCard]
  - [EmailAddress]
  - [FileExtensions]
  - [Url]
  - [Required]
  - [StringLength]
  - [RegularExpression(@"\d{5}")]
  - [Range(100, 900)]
  - [Compare("Password")]
    - 屬性值是否等於另一個屬性
    - 例如：輸入密碼 v.s. 確認密碼

使用 DataAnnotations 定義欄位驗證

All services from your imperative.

- 引用命名空間
  - System.Web.Mvc
- 欄位驗證屬性
  - [Remote("ActionName", "ControllerName")]
- 新增 **Remote** 屬性 (Convension over Configuration)
  - 必須搭配 Microsoft jQuery Unobtrusive Ajax 套件
  - 將輸入值透過 AJAX 送到指定的 Action 做驗證
    - `return Json(true, JsonRequestBehavior.AllowGet);`
    - `return Json("ERROR_MESSAGE", JsonRequestBehavior.AllowGet);`
- 新增 AllowHtml 屬性 (此屬性非欄位驗證)
  - 在 Controller 裡可用 `[ValidateInput(false)]`

繼承 DataTypeAttribute 自訂欄位驗證

All services from your imperative.

```
[AttributeUsage(AttributeTargets.Property, AllowMultiple = false)]
public sealed class EmailAddressAttribute : DataTypeAttribute
{
    private static Regex _regex = new Regex("PATTERN", RegexOptions.IgnoreCase);

    public EmailAddressAttribute() : base(DataType.EmailAddress)
    {
        ErrorMessage = MvcResources.EmailAddressAttribute_Invalid;
    }

    public override bool IsValid(object value)
    {
        if (value == null)
        {
            return true;
        }

        string valueAsString = value as string;
        return valueAsString != null && _regex.Match(valueAsString).Length > 0;
    }
}
```

繼承 ValidationAttribute 自訂欄位驗證

All services from your imperative.

```
[AttributeUsage(AttributeTargets.Property, AllowMultiple = false)]
public class MaxWordsAttribute : ValidationAttribute
{
    public MaxWordsAttribute(int maxWords) : base("{0} has too many words.")
    {
        _maxWords = maxWords;
    }

    protected override ValidationResult IsValid(
        object value, ValidationContext validationContext)
    {
        if (value != null)
        {
            var valueAsString = value.ToString();
            if (valueAsString.Split(' ').Length > _maxWords)
            {
                var errorMessage = FormatErrorMessage(validationContext.DisplayName);
                return new ValidationResult(errorMessage);
            }
        }
        return ValidationResult.Success;
    }

    private readonly int _maxWords;
}
```

在 Model 實作 IValidatableObject 介面

All services from your imperative.

□ 設計一個能夠自我驗證商業邏輯的 Model

```
public partial class Order : IValidatableObject
{
    public IEnumerable<ValidationResult> Validate(
                                        ValidationContext validationContext)
    {
        if (LastName != null &&
            LastName.Split(' ').Length > 10)
        {
            yield return new ValidationResult(
                "The last name has too many words!", new []{ "LastName" });
        }
    }

    // rest of Order implementation and properties
    // ...
}
```

All services from your imperative.

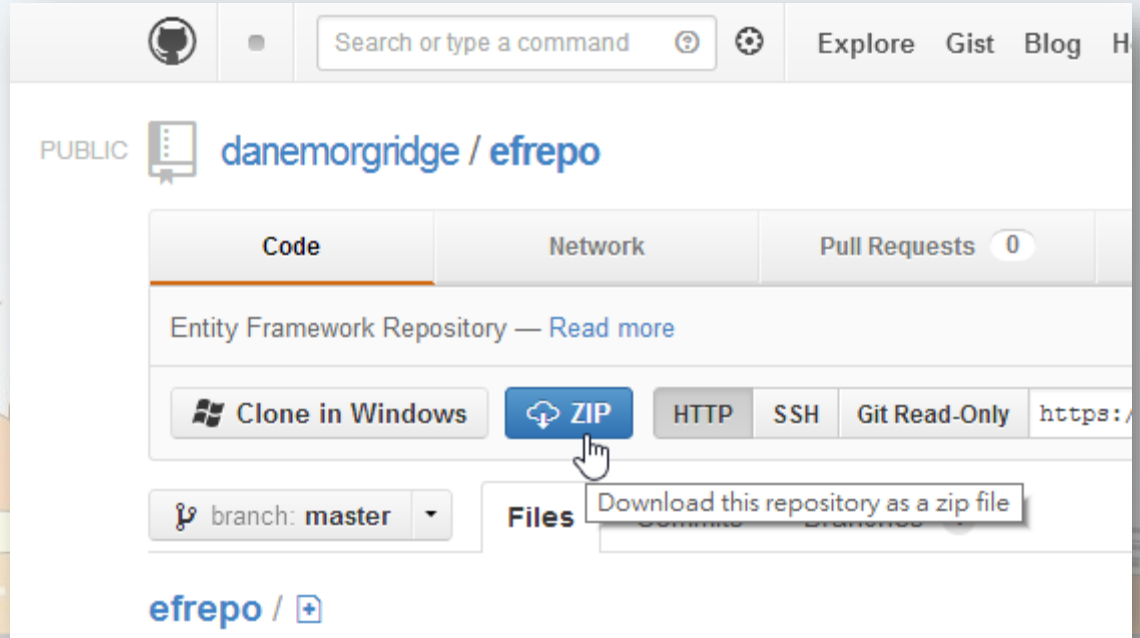
Code Generator

程式碼產生器實戰

資料倉儲 (Repository) 產生器

All services from your imperative.

- 常用產生器
  - Entity Framework Repository & Unit of Work T4 Template
    https://github.com/doggy8088/efrepo
- 實作 EFRepository
  - 下載：https://github.com/danemorgridge/efrepo

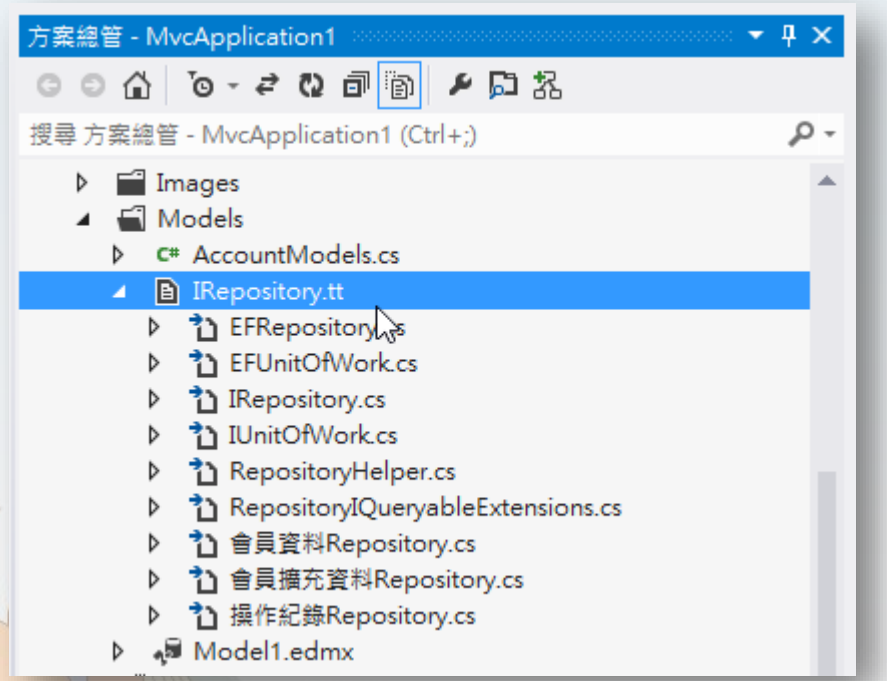


實作 EFRepository

All services from your imperative.

建置步驟

- 建立 Entity Framework 資料模型 (EDMX)
- 將 **DbContext Templates\IRepository.tt** 拖曳至 Models



ASP.NET MVC 5 開發實戰

控制器相關技術開發實務

多奇數位創意有限公司

技術總監 黃保翕 (Will 保哥)

部落格：http://blog.miniasp.com/

Routing

網址路由

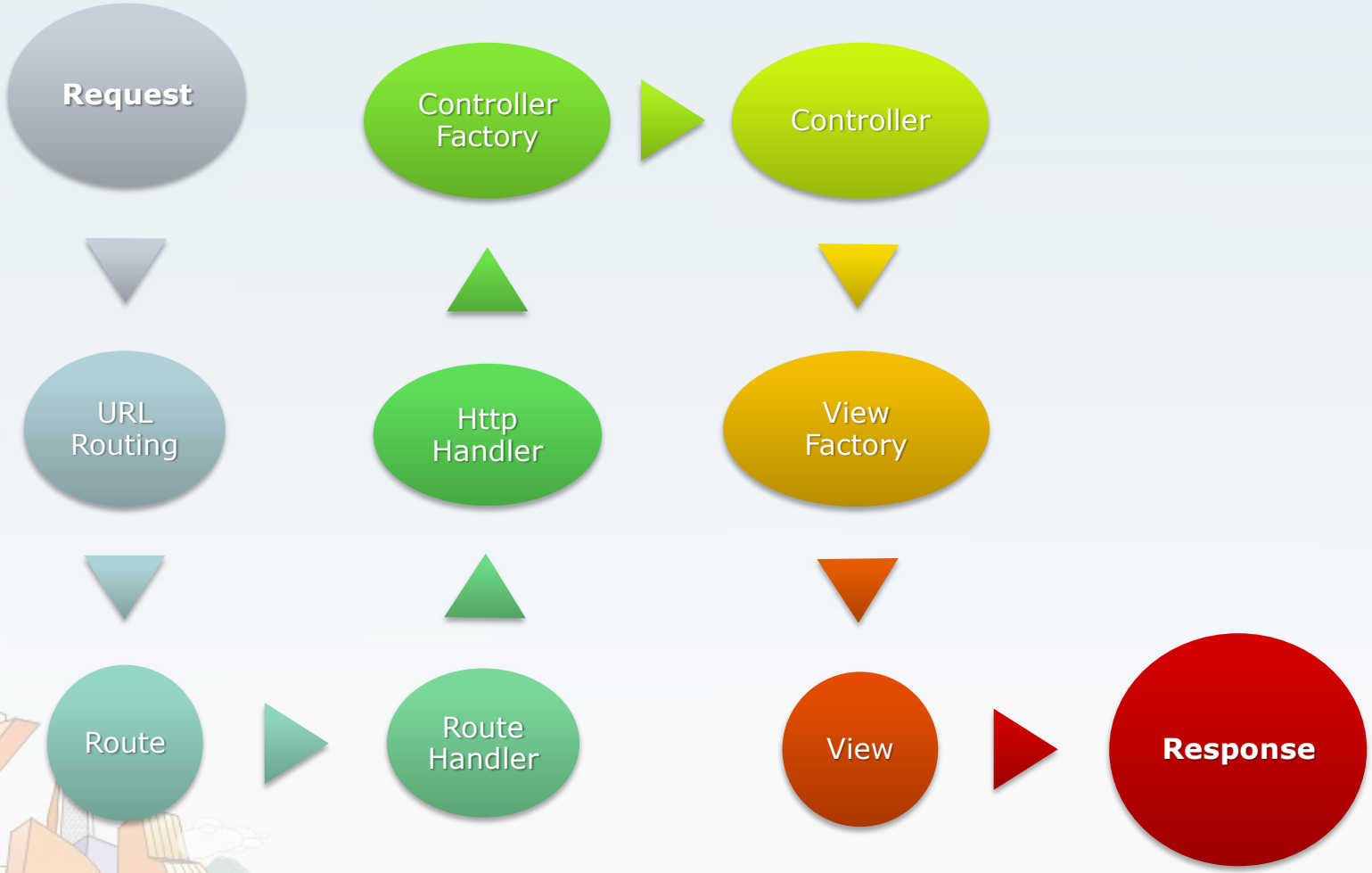
ASP.NET MVC 執行生命週期

All services from your imperative.

- 網址路由比對
  - 至少必須取得 {controller} 與 {action} 路由變數
- 執行 Controller 與 Action
  - 透過 Action 方法取得 ActionResult
- 執行 View 並回傳結果
  - 透過 RazorViewEngine 或 WebFormViewEngine

瞭解 ASP.NET MVC 執行生命週期

All services from your imperative.



網址路由 (Routing) 的任務

All services from your imperative.

- 比對透過瀏覽器傳來的 HTTP 要求
  - HTTP 要求網址
    - `http://localhost:6302/Home/About`
- 回應適當的網址給瀏覽器
  - Razor 語法
    - `@Html.ActionLink("關於", "About", "Home")`
  - HTML 輸出
    - `<a href="/Home/About">關於</a>`

瞭解 RouteConfig.cs 的內容

All services from your imperative.

- 全域物件
  - RouteTable.Routes
- Routing 集中定義
  - RouteConfig.RegisterRoutes
- 擴充方法
  - IgnoreRoute
    - 忽略路由，將要求交還給 IIS
    - 由 IIS 選擇適當的 HttpHandler 處理此要求
  - MapRoute
    - 對應路由，將要求交給 MvcHandler 處理

瞭解 RouteConfig.cs 的內容

All services from your imperative.

```
1 個參考
public class RouteConfig
{
    1 個參考
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new
            {
                controller = "Home",
                action = "Index",
                id = UrlParameter.Optional
            });
    }
}
```

IgnoreRoute 與 MapRoute 擴充方法

All services from your imperative.

- 網址列上可外加預設以外的參數
  - 明確列出路由參數名稱，例如：{id}
  - 必要路由參數：{controller} 與 {action}
- Catch All RouteValue
  - {*anything}
- 允許 RouteValue 傳入為 null
  - 使用 `UrlParameter.Optional` 型別
  - 因為 **匿名型別** 不允許**屬性值**為 **null**
- 主要限制
  - 不允許定義路由時出現查詢字串 (QueryString)

HTTP 要求的 URL 如何對應網址路由

All services from your imperative.

- 範例 1：
  - http://localhost/Trace.axd/a/b/c/d/e
- 比對順序：
  - 比對 routes.IgnoreRoute 的 {resource}.axd/{*pathInfo} 網址格式。
  - {resource}.axd 首先比對到 Trace.axd，因此繼續比對下一個 RouteValue。
  - 接著比對 {*pathInfo}，得到 a/b/c/d/e。
  - 因為所有的 RouteValue 都比對成功，所以這一次的 HTTP 要求會由此網址路由提供服務。
- 比對結果：
  - 該網址以 routes.IgnoreRoute 擴充方法進行比對成功，因此，Routing 模組不會將本次 HTTP 要求丟給 ASP.NET MVC 執行，而是將 HTTP 重新交還給 IIS 的其他 HTTP 模組處理此要求。

HTTP 要求的 URL 如何對應網址路由

All services from your imperative.

- 範例 2：
  - http://localhost/Member/Detail?id=123

```
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new {
                controller = "Home",
                action = "Index",
                id = UrlParameter.Optional
            }
        );
    }
}
```

HTTP 要求的 URL 如何對應網址路由

All services from your imperative.

- 範例 3：
  - http://localhost/Home/Cart.aspx?id=123

```
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new {
                controller = "Home",
                action = "Index",
                id = UrlParameter.Optional
            }
        );
    }
}
```

路由比對的注意事項

All services from your imperative.

- 只有 / 與 ? 是無法進行比對的
  - 網址列中出現以下字元會比對進變數
    - 小數點 (.)
    - 底線 (_)
    - 減號 (-)
  - 例如
    - http://api.jquery.com/**jQuery.each**/
    - http://localhost/**Docs/jQuery-each**
- 相關文章
  - http://bit.ly/134LNXm (.NET 4.0 / 4.5 的專案範本差異)
  - http://bit.ly/17thVQW (講解 URL 結構)
  - http://bit.ly/aspnetroutingskills (Routing 觀念與技巧)

替網址路由加上**路由值**的條件約束

All services from your imperative.

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new {
        controller = "Home",
        action = "Index",
        id = UrlParameter.Optional
    },
    // 加上路由值的條件約束，使用正則表達式(RegEx)
    constraints: new {
        id = @"\d+"
    }
);
```

屬性路由 (Attribute Routing)

All services from your imperative.

- ASP.NET MVC 5 內建屬性路由機制
  - 修改 App_Start\RouteConfig.cs
    - `routes.MapMvcAttributeRoutes();`
    - **通常會設定在 `routes.MapRoute()` 之前！**
  - 套用 Controller / Action 路由屬性
    - `[Route("products")]`
    - `[Route("products/{id}")]`
- 參考連結
  - Attribute Routing in ASP.NET MVC 5
    - http://bit.ly/attribute-routing-in-asp-net-mvc-5
  - Attribute Routing in Web API 2
    - http://bit.ly/attribute-routing-in-web-api-2

All services from your imperative.

Controller 與 Action

控制器與動作方法

關於 Controller 的特性

All services from your imperative.

- 控制器類別 (Controller Class)
  - Controller 必須為**公開類別**
  - Controller **名稱**必須以 **Controller 結尾**
  - 必須繼承自 ASP.NET MVC 內建的 Controller 類別
或繼承有實作 IController 介面的自訂類別
或直接實作 IController 介面
- 動作方法 (Action Method)
  - 所有動作方法必須為**公開方法**。
  - 任何**非公開**的方法如宣告為 private 或 protected 方法，都**不會**被視為一個動作方法。

實作 BaseController

All services from your imperative.

□ 共用輔助方法與 HandleUnknownAction

```
namespace MvcApplication1.Controllers
{
    public class BaseController : Controller
    {
        protected override void HandleUnknownAction(string actionName)
        {
            if (this.ControllerContext.HttpContext
                    .Request.HttpMethod.ToUpper() == "GET")
            {
                this.View(actionName).ExecuteResult(this.ControllerContext);
            }
            else
            {
                base.HandleUnknownAction(actionName);
            }
         }
    }
}
```

Action Result

動作結果

各種 ActionResult 類型

All services from your imperative.

- ASP.NET MVC 2
  - ViewResult
  - PartialViewResult
  - ContentResult
  - FileResult
  - JsonResult
  - JavaScriptResult
  - RedirectResult
  - EmptyResult
- ASP.NET MVC 3 / 4 / 5
  - HttpNotFoundResult
  - HttpStatusCodeResult
  - HttpNotFoundResult
  - HttpUnauthorizedResult
  - RedirectResult (更新版)

關於 Controller 類別的輔助方法

All services from your imperative.

- ASP.NET MVC 2
  - View
  - PartialView
  - Content
  - File
  - Json
  - JavaScript
  - Redirect、RedirectToRoute、RedirectToAction
- ASP.NET MVC 3 / 4 / 5
  - HttpNotFound
  - RedirectPermanent
  - RedirectToActionPermanent
  - RedirectToRoutePermanent

ViewResult

All services from your imperative.

- 回傳方式
  - return View();
  - return View("About2");
  - return View("Index", "**_Layout2**");
- 尋找 View 的順序
  - ~/Views/Home/AAA.aspx
  - ~/Views/Home/AAA.ascx
  - ~/Views/Shared/AAA.aspx
  - ~/Views/Shared/AAA.ascx
  - ~/Views/Home/AAA.cshtml
  - ~/Views/Home/AAA.vbhtml
  - ~/Views/Shared/AAA.cshtml
  - ~/Views/Shared/AAA.vbhtml

WebFormViewEngine

RazorViewEngine

ViewResult

All services from your imperative.

- 減少搜尋 View 檔案的時間
  - 請將以下程式碼加入 Global.asax.cs 檔案

```
ViewEngines.Engines.Clear();

ViewEngines.Engines.Add(new RazorViewEngine());

// ViewEngines.Engines.Add(new WebFormViewEngine());
```

PartialViewResult

All services from your imperative.

- 回傳方式
  - return PartialView();
  - **與 ViewResult 唯一的差異**：不會載入 _Layout 主版頁面
- 尋找 PartialView 的順序
  - ~/Views/Home/AAA.aspx
  - ~/Views/Home/AAA.ascx
  - ~/Views/Shared/AAA.aspx
  - ~/Views/Shared/AAA.ascx
  - ~/Views/Home/AAA.cshtml
  - ~/Views/Home/AAA.vbhtml
  - ~/Views/Shared/AAA.cshtml
  - ~/Views/Shared/AAA.vbhtml

ContentResult

All services from your imperative.

- 回傳方式
  - `return Content("<ROOT><TEXT>123</TEXT></ROOT>", "text/xml", System.Text.Encoding.UTF8);`
- 注意事項
  - 直接輸出
  - 如同使用 Response.Write
  - 千萬別再使用 Request 或 Response 物件
- 另一種使用方法
  - ```
    public string Robots() {
        return "Disallow: /";
    }
    ```
    ```

FileResult

All services from your imperative.

- 回傳方式
  - ```
    return File(
      Server.MapPath("~/App_Data/UserA/Avatar.png"),
      "image/png")
    ```
- 三種變形
  - FilePathResult：回應一個實體檔案的內容。
  - FileContentResult：回應一個 byte[] 內容。
  - FileStreamResult：回應一個 Stream 內容。
- 注意事項
  - 盡量不要指定**中文檔名**下載
  - Complete MIME Types List - FreeFormatter.com
    ```

使用中文檔名下載的密技

All services from your imperative.

```
public ActionResult GetFile()
{
    byte[] fileContent = GetFileByteArrayFromDB();

    if (Request.Browser.Browser == "IE" && Convert.ToInt32(Request.Browser.MajorVersion) < 9)
    {
        // 舊版 IE 使用舊的相容性作法
        return File(fileContent, "text/plain", Server.UrlPathEncode("你的報表.txt"));
    }
    else
    {
        // 新版瀏覽器使用RFC2231規範的Header Value作法
        return File(fileContent, "text/plain", "你的報表.txt");
    }
}
```

http://bit.ly/ChineseFileName

JavaScriptResult

All services from your imperative.

回傳方式

- `return JavaScript("alert('ok')");`
- `@Ajax.ActionLink("Run JavaScript", "JavaScript", new AjaxOptions())`
- 安裝 Microsoft jQuery Unobtrusive Ajax 套件
- App_Start\BundleConfig.cs

```
bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
    "~/Scripts/jquery-{version}.js",
    "~/Scripts/jquery.unobtrusive-ajax.js"));
```

這是動作名稱
(Action Name)

注意事項

- JavaScriptResult 預設的 Content-Type 為 application/x-javascript

JsonResult

All services from your imperative.

回傳方式

```
return Json(new {
        id = 1,
        name = "Will",
        CreatedOn = DateTime.Now
    }, JsonRequestBehavior.AllowGet);
```

輸出結果

```
{"id":1,"name":"Will","CreatedOn":"\/Date(1350116309992)\/"}
```

注意事項

- JsonResult 預設的 Content-Type 為 application/json

RedirectResult

All services from your imperative.

- 回傳方式
  - `return Redirect("/Home/About");`
- 三種變形
  - RedirectToAction
  - RedirectToActionPermanent
  - RedirectToRoute
  - RedirectToRoutePermanent
- 注意事項
  - HTTP 301 永久轉址
    - Redirection - SEO Best Practices - Moz
  - HTTP 302 暫時轉址

HttpStatusCodeResult

All services from your imperative.

- 回傳方式
  - `return new HttpStatusCodeResult(201, "資料已被成功建立");`
  - `return new HttpStatusCodeResult(HttpStatusCode.Created, "成功建立");`
  - `return HttpNotFound();`
  - `return new HttpUnauthorizedResult();`
- HTTP 狀態碼大致分成五種：
  - 1xx - 參考資訊 (Informational)。
  - 2xx - 成功 (OK)：一般最常見的HTTP狀態碼如200代表OK，也就是網頁正常回應的意思，201代表Created代表伺服器端已經成功建立資源。
  - 3xx - 重新導向 (Redirection)：剛剛看過的302代表Found，意即找到這個資源，但暫時移到另一個URL，而301則代表Moved Permanently，意即URL已經發生永久改變，用戶端必須轉向到另一個URL，且不用保留原本URL的紀錄。
  - 4xx - 用戶端錯誤 (Client Error)：這裡最常見的就是404 Not Found，代表找不到網頁，還有401 Unauthorized，代表拒絕存取也都是非常常見的用戶端錯誤。
  - 5xx - 伺服器錯誤 (Server Error)：當伺服器發生錯誤時會回應5xx的狀態碼，而500 Internal Server Error屬內部伺服器錯誤，也是經常看見的HTTP狀態碼。

Model Binding

模型繫結

如何從 Action 傳資料到 View 裡

All services from your imperative.

- ViewData.Model
  - 強型別傳值
- ViewData["KeyName"]
  - 弱型別傳值，使用 Dictionary 物件
- ViewBag.KeyName
  - 使用動態型別，骨子裡用的是 ViewData
- TempData ["KeyName"]
  - 弱型別傳值，使用 Dictionary 物件
  - 寫入的資料被讀過一次就會被刪除！
  - 預設會將資料儲存在 Session 裡！

如何從 View 傳資料到 Action 裡

All services from your imperative.

- 簡單模型繫結
  - 使用 C# 基礎型別取得表單欄位資料
  - 使用 FormCollection 取得表單所有欄位資料
  - 可在 Action 裡驗證模型繫結結果
- 複雜模型繫結
  - 使用任意 .NET 型別
  - 會在 Action 前預先判斷模型繫結的驗證結果

簡單模型繫結

All services from your imperative.

- DefaultModelBinder
  - int
  - bool
  - DateTime
  - string
  - …
- 使用 FormCollection 取得表單資料
  - 使用 NameValueCollection 型別接入變數

複雜模型繫結

All services from your imperative.

- DefaultModelBinder
  - 任何物件屬性都能繫結

```
public ActionResult Add(Product product)
{
    //
}
```

  - 可以繫結陣列物件

```
public ActionResult Add(IList<Product> products)
{
    //
}
```

判斷模型繫結的模型驗證結果

All services from your imperative.

- 模型繫結**預先驗證**
  - 自動完成所有模型繫結與模型驗證工作
    - ModelState.IsValid
- 模型繫結**延遲驗證**
  - 到 Action 裡才執行模型繫結與模型驗證工作
    - UpdateModel<T>
      - 驗證失敗就會丟出 InvalidOperationException 例外
    - TryUpdateModel<T>
      - 不用再額外判斷 ModelState.IsValid 的值!
- 自訂**驗證失敗錯誤訊息**到 View 的方法
  - ModelState.AddModelError("欄位名稱", "錯誤訊息")

使用弱型別過濾不允許傳入的屬性

All services from your imperative.

- 模型繫結**預先驗證**
  - 負向表列 (黑名單機制)
    - Bind(Exclude="ID,CreatedOn")
  - 正向表列 (白名單機制)
    - Bind(Include="Name,Tel")
  - 套用 Bind 屬性的地方
    - 實體模型 (Entity Model)
    - 動作方法參數 (Action Method's Parameters)
- 模型繫結**延遲驗證**

```
TryUpdateModel(user, includeProperties: new[] { "FirstName" });
```

使用強型別過濾不允許傳入的屬性

All services from your imperative.

- 模型繫結**預先驗證**
  - 負向表列 (黑名單機制) (標示 `[ReadOnly(true)]` 代表不繫結)

```
public class User {
    public string FirstName { get; set; }
    [ReadOnly(true)]
    public bool IsAdmin { get; set; }
}
```

  - 強型別的預先驗證沒有「白名單」機制
- 模型繫結**延遲驗證**
  - 透過**介面**限制傳入參數的數量 (白名單機制)
    - `TryUpdateModel<IUserInputModel>(user);`
    - 範例：http://bit.ly/YvHnnx
    - 用**介面**做**模型繫結**並不會影響原本的**輸入驗證**與**模型驗證**

Action Filter

動作過濾器

動作名稱選擇器

All services from your imperative.

- ActionNameSelectorAttribute
  - [ActionName("About")]
- 注意事項
  - 這不是「動作過濾器」
  - 但用法類似，一樣套用在 Action 上面

動作方法選擇器

All services from your imperative.

- ActionMethodSelectorAttribute
  - [NonAction]
  - [AcceptVerbs("POST")]
  - [HttpGet] / [HttpPost] / [HttpDelete] / [HttpPut]
  - [HttpHead] / [HttpOptions] / [HttpPatch]
  - [AjaxOnly]　　來自 MVC Futures 套件(NuGet)

    ```
    Install-Package Microsoft.AspNet.Mvc.Futures -Version 5.0.0
    ```

    https://www.nuget.org/packages/Microsoft.AspNet.Mvc.Futures/5.0.0

- 注意事項
  - 這不是「動作過濾器」
  - 但用法類似，一樣套用在 Action 上面

動作過濾器類型

All services from your imperative.

- 驗證過濾器 (IAuthenticationFilter)
  - OnAuthentication
  - OnAuthenticationChallenge
- 授權過濾器 (IAuthorizationFilter)
  - OnAuthorization
- 動作過濾器 (IActionFilter)
  - OnActionExecuting
  - OnActionExecuted
- 結果過濾器 (IResultFilter)
  - OnResultExecuting
  - OnResultExecuted
- 例外過濾器 (IExceptionFilter)
  - OnException

動作過濾器的類別繼承關係與介面實作

All services from your imperative.

IMvcFilter

- FilterAttribute
  - ActionFilterAttribute (IActionFilter, IResultFilter)
    - ContentTypeAttribute
    - CopyAsyncParametersAttribute
    - AsyncTimeoutAttribute
    - OutputCacheAttribute (IExceptionFilter)
  - HandleErrorAttribute (IExceptionFilter)
  - AuthorizeAttribute (IAuthorizationFilter)
  - ChildActionOnlyAttribute (IAuthorizationFilter)
  - RequireHttpsAttribute (IAuthorizationFilter)
  - ValidateAntiForgeryTokenAttribute (IAuthorizationFilter)
  - ValidateInputAttribute (IAuthorizationFilter)

身分驗證過濾器 (Authentication filters)

All services from your imperative.

- ASP.NET MVC 5 新增的動作過濾器類型
  - **動作過濾器**要繼承 **ActionFilterAttribute** 類別
  - **認證過濾器**要實作 **IAuthenticationFilter** 介面
    - **OnAuthentication**
      (AuthenticationContext filterContext)
    - **OnAuthenticationChallenge**
      (AuthenticationChallengeContext filterContext)
- 更精細的進行身分驗證機制
  - 依據特定 Action
  - 依據特定 Controller
  - 全站套用 (All Controllers)

範例程式

All services from your imperative.

```
1 個參考
public class BasicAuthAttribute : ActionFilterAttribute, IAuthenticationFilter
{
    private bool _authenticate;

    0 個參考
    public void OnAuthentication(AuthenticationContext filterContext)
    {
        _authenticate = (filterContext.ActionDescriptor.GetCustomAttributes(
            typeof(OverrideAuthenticationAttribute), true).Length == 0);
    }

    0 個參考
    public void OnAuthenticationChallenge(AuthenticationChallengeContext filterContext)
    {
        var user = filterContext.HttpContext.User;
        if (user == null || !user.Identity.IsAuthenticated)
        {
            filterContext.Result = new HttpUnauthorizedResult();
        }
    }
}
```

授權過濾器

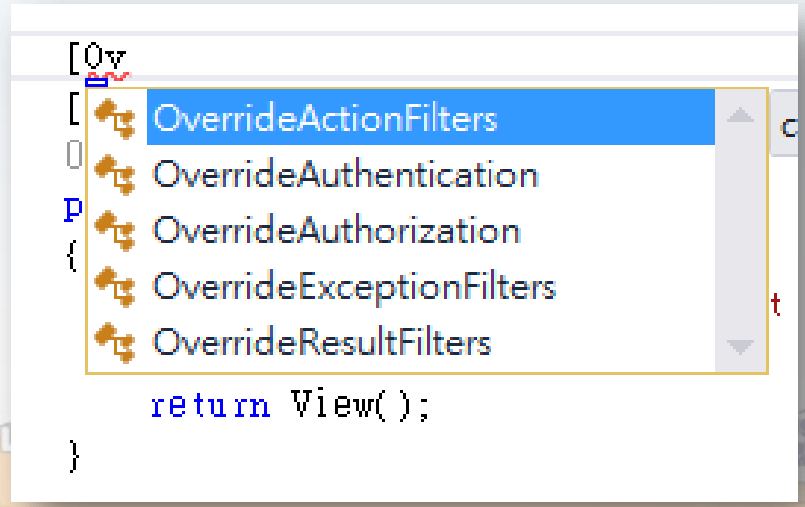
All services from your imperative.

- 授權控制
  - [Authorize]
    - [AllowAnonymous] (這不是動作過濾器)
- 安全網頁
  - [RequireHttps]
- 輸入驗證
  - [ValidateInput]
  - [ValidateAntiForgeryToken]
- 動作篩選
  - [ChildActionOnly]

動作過濾器覆寫 (Filter overrides)

All services from your imperative.

- ASP.NET MVC 4 之前
  - 你可以將動作過濾器定義在全域 (全站套用)
  - 無法針對 Action 或 Controller 取消特定類型的過濾器
- ASP.NET MVC 5 之後
  - 可以針對 Action 或 Controller 取消特定類型的過濾器
- ASP.NET MVC 5 新增的動作過濾器
  - OverrideActionFilters
  - OverrideAuthentication
  - OverrideAuthorization
  - OverrideExceptionFilters
  - OverrideResultFilters



動作過濾器 & 結果過濾器

All services from your imperative.

- 非同步控制器 (已過時)
  - [AsyncTimeout]
  - [NoAsyncTimeout]
- 輸出快取
  - [OutputCache]　　(IExceptionFilter)

例外過濾器

All services from your imperative.

- 錯誤處理
  - `[HandleError]`
  - `[HandleError(View="Error2", ExceptionType=typeof(ArgumentException))]`
- 注意事項
  - **web.config 設定正確，該屬性才會生效**
    - <customErrors mode="On" />
    - <customErrors mode="RemoteOnly" />
  - **預設錯誤頁面 (ViewName = "Error")**
    - /Views/[Controller]/Error.cshtml
    - /Views/Shared/Error.cshtml
- 在 ASP.NET MVC 4 / 5 套用 ELMAH 套件
  - http://bit.ly/mvc4-handleerror-elmah

載入全站共用的 Action Filter

All services from your imperative.

- 檔案路徑
  - Global.asax.cs
  - App_Start\FilterConfig.cs
- 方法名稱
  - RegisterGlobalFilters
- 全域物件
  - GlobalFilters.Filters (GlobalFilterCollection)

```
public class FilterConfig
{
    public static void RegisterGlobalFilters(GlobalFilterCollection filters)
    {
        filters.Add(new HandleErrorAttribute());
    }
}
```

自訂過濾器的設計方法

All services from your imperative.

- 繼承 ActionFilterAttribute 屬性類別
  - OnActionExecuting
  - OnActionExecuted
  - OnResultExecuting
  - OnResultExecuted
- 繼承至現有的 ActionFilterAttribute 類別
  - [Authorize]
  - [HandleError]

ASP.NET MVC 5 開發實戰

檢視頁面相關技術實務

多奇數位創意有限公司

技術總監 黃保翕 (Will 保哥)

部落格：http://blog.miniasp.com/

All services from your imperative.

Razor Syntax

RAZOR 語法速成

RazorView vs. WebFormView

All services from your imperative.

- WebFormView
  - `<%= Html.Encode(DateTime.Now) %>`
  - `<%: DateTime.Now %>`
  - *.aspx / *.ascx / *.master
- RazorView
  - @DateTime.Now
  - *.cshtml / *.vbhtml

Razor 並不是一個程式語言，
它只是一種用在View頁面的程式碼區塊**撰寫風格**罷了！

Razor 基本語法

All services from your imperative.

語法範例

```
<p>
   現在時刻: @DateTime.Now
</p>
```

```
<p>
   會員名稱: @(User.Identity.Name + Model.MemberLevel)   @(i++)
   啟用狀態: @(ViewBag.IsEnabled ? "啟用" : "停用")
</p>
```

```
@{
     var name = "Will";
     var message = "你好，我是 " + name;
}
```

```
@{
    var name = "Will";
    @:你好，我是 @name
}
```

```
@@Will_Huang
```

```
@*<hgroup class="title">
    <h1>@ViewBag.Title.</h1>
    <h2>@ViewBag.Message</h2>
</hgroup>*@
```

Razor 與 HTML 混合輸出

All services from your imperative.

語法範例

```
@if (ViewBag.IsEnabled) {
    @:啟用
} else {
    @:停用
}
```

如果ViewBag參數並不存在，其回傳的結果將為null，但是 if 判斷式中必須回傳 Boolean 型別，因此會引發「無法將 null 轉換成 'bool'，因為它是不可為 null 的實值型別」例外狀況。

```
@if (ViewBag.IsEnabled != null && ViewBag.IsEnabled) {
    @:啟用
} else {
    @:停用
}
```

Razor 與 HTML 混合輸出

All services from your imperative.

語法範例	

`@if (ViewBag.IsEnabled)`
`{`
`  @:啟用`
`}`
`else`
`{`
`  @:停用`
`}`
	`@if (ViewBag.IsEnabled)`
	`{`
	`  啟用`
	`}`
	`else`
	`{`
	`  停用`
	`}`

`@if (ViewBag.IsEnabled)`
`{`
`  <span>啟用</span>`
`}`
`else`
`{`
`  <span>停用</span>`
`}`
	`@if (ViewBag.IsEnabled) {`
	`  <text>`
	`  顯示啟用的HTML段落：`
	`  <p>`
	`  @ViewBag.EnabledMessage`
	`  </p>`
	`  </text>`
	`}`

Razor 與 HTML 混合輸出

All services from your imperative.

語法範例

```
@{
    var is_valid = true;

    if(!is_valid){
        <text>無權限!</text>
    }
}
```

```
<ul>
@foreach (var myItem in Request.ServerVariables) {
    <li>@myItem</li>
}
</ul>
```

```
@{
    <h3>團隊成員</h3> string[] teamMembers = {"Will", "James", "Brinkley", "Porin"};

    foreach (var person in teamMembers)
    {
        <p>@person</p>
    }
}
```

Razor 與 HTML 混合輸出陷阱與技巧

All services from your imperative.

語法範例	
@ViewBag.Name先生	您好，@ViewBag.Name先生
@(ViewBag.Name)先生	您好，@ViewBag.Name先生
@ViewBag.MvcNamespace.Controller	@(ViewBag.MvcNamespace).Controller
命名空間是 @(ViewBag.MvcNamespace).	命名空間是 @ViewBag.MvcNamespace.
teacher@example.com	
teacher@@example.com	
The Will Will Web	

Razor 與 HTML 混合輸出陷阱與技巧

All services from your imperative.

語法範例

```
<a id="LinkBlog01" href="http://blog.miniasp.com">The Will Will Web</a>
```

```
<a id="Link@(ViewBag.BlogID)" href="http://blog.miniasp.com">The Will Will Web</a>
```

```
@{
    ViewBag.Description = "<span style='font-weight: bold;'>描述文字</span>";
}

@ViewBag.Description
```

```
@{
    ViewBag.Description = "<span style='font-weight: bold;'>這是描述文字</span>";
}

@Html.Raw(ViewBag.Description)
```

可愛的 @

All services from your imperative.

Visual Studio 2013 非常的有“智慧”

- @ViewBag.Subject
- This is @ViewBag.Subject.
- @(ViewBag.Subject).OK?
- teacher@example.com
- id_@(ViewBag.ID)
- @@willh

- @:目前是第 @i 個迴圈

Layout

主版頁面

Razor 的主版頁面框架

All services from your imperative.

- Razor 頁面執行順序
  - /Views/_ViewStart.cshtml
    - 預設會先載入 _ViewStart.cshtml
  - /Views/Home/Index.cshtml
    - 然後開始執行這個 Action 指定的 View
  - /Views/Shared/_Layout.cshtml
    - 最後載入 Layout 頁面，並將主要內容填入!

關於 _ViewStart.cshtml

All services from your imperative.

- 關於 _ViewStart.cshtml 的特性
  - 此 View 會比原本要執行的 View 先執行
  - 在 /Views/ 裡每層目錄都能有這個檔案
  - _ViewStart.cshtml 執行順序
    - /Views/ 根目錄下的 _ViewStart.cshtml (如果有的話)
    - 再執行往下一層目錄裡的 _ViewStart.cshtml (如果有的話)
    - 最後才會執行原本該執行的 View 頁面 (*.cshtml)

Layout 頁面的 @ 語法

All services from your imperative.

- 主版頁面 (/Views/Shared/_Layout.cshtml)
  - 類似以前的 MasterPage 在頁面中挖洞！
  - @RenderSection("footer", required:false)
  - @RenderBody()
  - @if (!IsSectionDefined("footer")){
    }
- 內容頁面 (/Views/*/*.cshtml)
  - @section footer {
    ……
    }

All services from your imperative.

Razor Syntax - Advanced

RAZOR 語法進階

@helper 輔助方法

All services from your imperative.

- 基本語法

HomeController.cs | About.cshtml

用戶端物件和事件 | (沒有事件)

```
1  @{
2      ViewBag.Title = "About Us";
3  }
4
5  @helper ToLink(string url)
6  {
7      <a href="@url">@url</a>
8  }
9
10 <h2>ASP.NET Web Pages Helper Sample</h2>
11 <p>
12     我是這的部落格: @ToLink("http://blog.miniasp.com")
13 </p>
14
```

- 共用全站的 @helper
  - App_Code*ClassName*.cshtml

輸出 HTML 或輸出 HTML String

All services from your imperative.

```
@helper OutputAddress(string ZipCode)
{
   <p>
      One Microsoft Way<br />
      Redmond, WA @ZipCode
   </p>
}

@OutputAddress("98052-6399")
@OutputAddress("98052-6399").ToHtmlString()

@Html.Raw(Utility.OutputAddress("98052-6399").ToHtmlString())
@Utility.OutputAddress("98052-6399").ToHtmlString()
```

設計給全站使用的 @helper

All services from your imperative.

- 建立 App_Code 目錄
- 在 App_Code 目錄裡建立 Razor View 頁面
  - 例如：MyHelper.cshtml

```
@helper ToLink(string link) {
    <a href="@link">@link</a>
}
```

- 使用方式
  - **@檔名.方法名稱**
  - 例如：
    @MyHelper.ToLink("http://blog.miniasp.com")

@functions 自訂函式

All services from your imperative.

- 基本語法

```
@functions {
    public IHtmlString GetYesterday() {
        var theDay = DateTime.Now.AddDays(-1);
        return new HtmlString(theDay.ToShortDateString());
    }
}
```

- 共用全站的 @functions
  - App_Code*ClassName*.cshtml
  - 移過去之後的方法或屬性，必須要宣告為靜態 (static) 才能讓各頁面取用

自訂 @functions 範例

All services from your imperative.

AtFunctions.cshtml

用戶端物件和事件 (沒有事件

```
1  @using System.Text 19
2  @{
3      ViewBag.Title = "About @functions";
4  }
5  @functions {
6      public DateTime TEST { get; set; }
7      public IHtmlString Go()
8      {
9          TEST = DateTime.Today;
10         return new HtmlString("<b>Go Go Go ...</b>");
11     }
12 }
13
14 <h2>About @@functions</h2>
15 <p>
16     @Go()
17     <hr />
18     @TEST
19 </p>
```

@model 引用參考資料型別 (強型別)

All services from your imperative.

- 基本語法

```
@model IEnumerable<MvcApplication6.Models.Product>
```

- 注意事項
  - @model 語法的第一個字是小寫字母的 m
  - 千萬不要跟 @Model 給搞混了

@using 引用命名空間

All services from your imperative.

- 基本語法

```
@using MvcApplication6.Models
@model IEnumerable<Product>
```

- 全站共用的引用命名空間
  - 調整 Views\web.config 設定檔

```
<pages pageBaseType="System.Web.Mvc.WebViewPage">
  <namespaces>
    <add namespace="System.Web.Mvc" />
    <add namespace="System.Web.Mvc.Ajax" />
    <add namespace="System.Web.Mvc.Html" />
    <add namespace="System.Web.Optimization"/>
    <add namespace="System.Web.Routing" />
    <add namespace="MvcApplication6.Models" />
  </namespaces>
</pages>
```

關於 /Views/web.config 設定檔

All services from your imperative.

- Razor 使用專屬的設定區段
  - 設定 pageBaseType
  - 預先載入命名空間！

```
<system.web.webPages.razor>
  <host factoryType="System.Web.Mvc.MvcWebRazorHostFactory, System.Web
  <pages pageBaseType="System.Web.Mvc.WebViewPage">
    <namespaces>
      <add namespace="System.Web.Mvc" />
      <add namespace="System.Web.Mvc.Ajax" />
      <add namespace="System.Web.Mvc.Html" />
      <add namespace="System.Web.Routing" />
    </namespaces>
  </pages>
</system.web.webPages.razor>
```

在 View 裡常用的物件 – 內容部分

All services from your imperative.

- @this.
  - @Layout , @RenderSection , @RenderBody
    - @section
  - @Url , @Html , @Ajax
  - @Model
  - @ViewBag , @ViewData , @TempData
  - @IsPost , @IsAjax
  - @ViewContext
  - @App , @Session , @Profile , @User
  - @Request , @Response

All services from your imperative.

HTML Helper & URL Helper

HTML 與 URL 輔助方法

關於 @Html 與 @Url 與 @Ajax Helper

All services from your imperative.

- 表單
  - @Html.BeginForm v.s. @Ajax.BeginForm
- 載入其他頁面
  - @Html.Action 與 @Html.RenderAction
  - @Html.Partial 與 @Html.RenderPartial
- 產生超連結
  - @Html.ActionLink 與 @Html.RouteLink
- 產生網址
  - @Url.Action 與 @Url.Content 與 @Url.RouteUrl

@Html.Action 與 @Html.RenderAction

All services from your imperative.

- ASP.NET MVC 2
  - `<%= Html.RenderAction("ActionName"); %>`
- ASP.NET MVC 3+ (Razor)
  - `@{ Html.RenderAction("ActionName"); }`
  - `@Html.Action("ActionName")`

使用HTML輔助方法載入部分檢視

All services from your imperative.

- 什麼是部分檢視
  - 沒有 Layout 的 View
- 有哪些用法？
  - Html.Partial("ajaxPage")
  - Html.Partial("ajaxPage", Model)
  - Html.Partial("ajaxPage", ViewData["Model"])
  - Html.Partial("ajaxPage", Model, ViewData["Model"])
- 範例
  - `@Html.Partial("OnlineUserCounter")`
  - `@Html.Partial("~/Views/Home/Online.cshtml")`
  - `@Html.Partial("../Home/Online")`

使用 HTML 輔助方法輸出超連結

All services from your imperative.

語法範例
`@Html.ActionLink("連結文字","ActionName")`
`@Html.ActionLink("連結文字", "ActionName", "ControllerName")`
`@Html.ActionLink("連結文字", "ActionName", new { id = 123, page = 5 })`
`@Html.ActionLink("連結文字", "ActionName", null, new { @class = "btnLink" })`
注意：**“連結文字”** 只要傳入**空白**或**null**，就會拋出 The Value cannot be null or empty. 例外狀況。
`<a href="/Home/Index" class="lnkButton"></a>`
`<a href="@Url.Action("ActionName")" class="lnkButton"></a>`

使用 @Html.ActionLink 的陷阱

All services from your imperative.

- 使用 @Html.ActionLink 所發生的詭異現象

http://bit.ly/ActionLinkMistake

```
av>
  <ul id="menu">
      <li>@Html.ActionLink("首頁", "Index", "Home", new { @class = "link1" })</li>
      <li>@Html.ActionLink("關於", "About", "Home")</li>
  </ul>
nav>
```

```
<nav>
    <ul id="menu">
        <li><a class="link1" href="/?Length=4">首頁</a></li>
        <li><a href="/Home/About">關於</a></li>
    </ul>
</nav>
```

URL 輔助方法

All services from your imperative.

用法	輸出結果
@Url.Action("About")	/Home/About
@Url.Action("About", new { id = 1 })	/Home/About/1
@Url.Action("About", "Home")	/Home/About
@Url.Action("About", "Home", new { PageNo = 1 })	/Home/About?PageNo=1
@Url.RouteUrl(new { id = 123 })	/Home/HelperSample/123
@Url.RouteUrl("路由名稱")	/root/Home/HelperSample
@Url.RouteUrl("路由名稱", new { id = 123 })	/root/Home/HelperSample/123
@Url.RouteUrl("路由名稱", new { id = 1 }, "https")	https://localhost/root/Home/HelperSample/1
@Url.RouteUrl(new { id = 123 })	/Home/HelperSample/123
@Url.Content("~/images/Logo.png")	/App1/images/Logo.png

顯示表單與資料

產生表單元素

All services from your imperative.

Html.BeginForm 語法

```
@using (Html.BeginForm("About", "Home")) {
    @Html.TextArea("Date")
    @Html.TextArea("MEMO")
    <input type="submit" />
}
```

檔案上傳

```
@using (Html.BeginForm("Upload", "File", FormMethod.Post,
    new { enctype = "multipart/form-data" }))
{
    @Html.TextBox("File1", "", new { type="file", size="25" })
    <input type="submit" />
}
```

顯示表單欄位的方法

All services from your imperative.

- 弱型別
  - @Html.Label("FieldName")
    - [DisplayName("Field Title")]
      public int FieldName { get; set; }
  - @Html.Display("FieldName")
- 強型別 (Templated Helpers)
  - @Html.EditorForModel()
  - @Html.EditorFor(m => m.Title)
  - @Html.LabelFor(m => m.Title)

弱型別欄位輸出的 Helper

All services from your imperative.

- 表單欄位
  - @Html.Hidden
  - @Html.TextBox
  - @Html.TextArea
  - @Html.Password
  - @Html.RadioButton
  - @Html.CheckBox
  - @Html.DropDownList
  - @Html.ListBox
- 顯示文字
  - @Html.Label
  - @Html.DisplayText
  - @Html.ValidationMessage
- 顯示所有 ModelState 錯誤
  - @Html.ValidationSummary(true)

常用表單輸入欄位 (弱型別)

All services from your imperative.

HTML 輔助方法	說明
Html.BeginForm()	輸出 <form> 標籤
Html.EndForm()	輸出 </form> 標籤
Html.Label()	輸出 <label> 標籤
Html.TextBox()	輸出 <input type="text"> 標籤
Html.TextArea()	輸出 <textarea> 標籤
Html.Password()	輸出 <input type="password"> 標籤
Html.Checkbox()	輸出 <input type="checkbox"> 標籤
Html.RadioButton()	輸出 <input type="radio"> 標籤
Html.DropDownList()	輸出 <select> 標籤
Html.ListBox()	輸出 <select multiple> 標籤
Html.Hidden()	輸出 <input type="hidden"> 標籤

關於 Html.DropDownList()

All services from your imperative.

□ 使用 SelectList 與 SelectListItem 類別

```
public ActionResult HelperSample()
{
    List<SelectListItem> listItem = new List<SelectListItem>();
    listItem.Add(new SelectListItem { Text = "是", Value = "1" });
    listItem.Add(new SelectListItem { Text = "否", Value = "0" });
    ViewData["List"] = new SelectList(listItem, "Value", "Text", "");
    return View("Test");
}
```

```
@Html.DropDownList("List", "請選擇")
```

關於 Html.Checkbox()

All services from your imperative.

□ 使用 Checkbox 要注意會輸出兩個欄位

View:

```
@Html.CheckBox("batch[" + i + "].IsConfirmDelete")
@Html.Hidden("batch[" + i + "].ClientId", data[i].ClientId)

@Html.EditorFor(modelItem => data[i].FirstName)
```

Controller:

```
public ActionResult BatchDelete(ClientBatchDeleteVM[] data)
public ActionResult BatchDelete(Client[] data)
public ActionResult BatchDelete(ClientBatchDeleteVM[] batch, Client[] data)
```

常用表單欄位屬性 (弱型別)

All services from your imperative.

HTML輔助方法	說明
Html.Id()	輸出特定欄位的id值
Html.Name()	輸出特定欄位的name值
Html.Value()	輸出特定欄位的value值
Html.ValidationSummary()	輸出表單驗證失敗時的錯誤訊息摘要

關於 Html.Value()

這裡的資料來自於 ASP.NET MVC 透過 ValueProvider 提供的值，可能是上一頁表單傳來的欄位資料、路由值、Query String 或是其他模型繫結進來的資料等等。

強型別欄位輸出

All services from your imperative.

- 表單欄位
  - @Html.HiddenFor
  - @Html.TextBoxFor
  - @Html.TextAreaFor
  - @Html.PasswordFor
  - @Html.RadioButtonFor
  - @Html.CheckBoxFor
  - @Html.DropDownListFor
  - @Html.ListBoxFor
  - @Html.EnumDropDownListFor
- 顯示文字
  - @Html.LabelFor
  - @Html.DisplayTextFor
  - @Html.ValidationMessageFor

範例程式

http://bit.ly/EnumSample

常用表單輸入欄位 (強型別)

All services from your imperative.

強型別輔助方法	說明
`Html.LabelFor()`	輸出 `<label>` 標籤
`Html.TextBoxFor()`	輸出 `<input type="text">` 標籤
`Html.TextAreaFor()`	輸出 `<textarea>` 標籤
`Html.PasswordFor()`	輸出 `<input type="password">` 標籤
`Html.CheckboxFor()`	輸出 `<input type="checkbox">` 標籤
`Html.RadioButtonFor()`	輸出 `<input type="radio">` 標籤
`Html.DropDownListFor()`	輸出 `<select>` 標籤
`Html.ListBoxFor()`	輸出 `<select multiple>` 標籤
`Html.HiddenFor()`	輸出 `<input type="hidden">` 標籤

常用表單欄位屬性 (強型別)

All services from your imperative.

HTML輔助方法	說明
Html.IdFor()	輸出資料模型相對應的欄位id值
Html.NameFor()	輸出資料模型相對應的欄位name值
Html.ValueFor()	輸出資料模型相對應的欄位value值。
Html.ValidationSummaryFor()	輸出表單驗證失敗時的錯誤訊息摘要
Html.DisplayNameFor()	顯示資料模型在Metadata定義的顯示名稱
Html.DisplayTextFor()	顯示資料模型的文字資料

關於 Html.Value()

這裡的資料來自於 ASP.NET MVC 透過 ValueProvider 提供的值，可能是上一頁表單傳來的欄位資料、路由值、Query String 或是其他模型繫結進來的資料等等。

強型別欄位輸出 For Model

All services from your imperative.

- 顯示整個 Model 的內容
  - @Html.DisplayForModel()
    - /Views/*ControllerName*/DisplayTemplates
    - /Views/Shared/DisplayTemplates
- 顯示整個 Model 的表單
  - @Html.EditorForModel()
    - /Views/*ControllerName*/EditorTemplates
    - /Views/Shared/EditorTemplates
- 顯示 **Model 定義的顯示名稱**
  - @Html.LabelForModel()

```
[DisplayName("客戶資料")]
public partial class 客戶
```

模擬各種 HTTP 動詞

All services from your imperative.

HTML 表單無法送出 DELETE 動詞

```
@using (Html.BeginForm("DeleteMessage", "Message"))
{
    @Html.HttpMethodOverride(HttpVerbs.Delete)
    @Html.Hidden("ID")
    <input type="submit" value="刪除此筆資料" />
}
```

Templated View & Templated Helpers

範本檢視的開發技巧

ASP.NET MVC 顯示資料

All services from your imperative.

- 透過字串指定欄位名稱
  - 資料來自於 Model 或 ViewData
  - @Html.Display("PropertyName")
- 透過 Lambda 表示法指定欄位
  - @Html.DisplayFor(model => model.PropertyName)
- 預設讀取 ViewData.Model 的方法
  - @Html.DisplayForModel()
  - @Html.DisplayFor(model => model)

ASP.NET MVC 編輯資料

All services from your imperative.

- 透過字串指定欄位名稱
  - 資料來自於 Model 或 ViewData
  - @Html.Editor("PropertyName")
- 透過 Lambda 表示法指定欄位
  - @Html.EditorFor(model => model.PropertyName)
- 預設讀取 ViewData.Model 的方法
  - @Html.EditorForModel()
  - @Html.EditorFor(model => model)

ASP.NET MVC 範本檢視 (Template View)

All services from your imperative.

- 顯示範本的方法
  - Html.DisplayFor
  - Html.DisplayForModel
  - Html.EditorFor
  - Html.EditorForModel
- 範本所在路徑
  - /Views/*ControllerName*/DisplayTemplates
  - /Views/Shared/DisplayTemplates
  - /Views/*ControllerName*/EditorTemplates
  - /Views/Shared/EditorTemplates

ASP.NET MVC 內建的 Templates

All services from your imperative.

- 內建的 Display Templates 有 10 個
  - DataType
    - EmailAddress
    - HiddenInput
    - Html
    - Text
    - Url
    - Collection
  - .NET Type
    - Boolean
    - Decimal
    - String
    - Object

ASP.NET MVC 內建的 Templates

All services from your imperative.

- 內建的 Editor Templates 有 22 個
  - DataType
    - HiddenInput、MultilineText、Password、Text、Collection、PhoneNumber、Url、EmailAddress
  - .NET Type
    - DateTime、Date、Time
    - Int32、UInt32、Int64、UInt64
    - Byte、SByte、Color、Boolean、Decimal、Object
    - String
      - 這是預設所有其他型別的範本！
      - 覆寫這個範本要注意不能使用 @model 強型別範本檢視

DateTime.cshtml

All services from your imperative.

□ 修正中文日期問題的解套方案

```
@model DateTime?


@if (Model.HasValue)
{
    @Html.TextBox("", Model.Value.ToString("yyyy/MM/dd"),
       new { @class = "text datebox", style = "width:120px" })
}
else
{
    @Html.TextBox("", "", new { @class = "text datebox", style = "width:120px" })
}
```

ASP.NET MVC 挑選範本檢視

All services from your imperative.

- 選用範本檢視的判斷順序
  - 範本檢視 (Templated View)
    - @Html.DisplayFor(m => m.IsAllow, "YesNo")
    - @Html.EditorFor(m => m.IsAllow, "YesNo")
  - UIHint 屬性
    - `using System.ComponentModel.DataAnnotations;`
    - `[UIHint("YesNo")]`
  - DataType 屬性
    - `using System.ComponentModel.DataAnnotations;`
    - `[DataType(DataType.EmailAddress)]`
  - 物件的資料型別名稱 (.NET 完整型別名稱)
    - `DateTime , Decimal , Boolean , Int32 , String`

ASP.NET MVC 5.1 增強功能

All services from your imperative.

- 強化 @Html.EditorFor() 輔助方法
  - 可透過 htmlAttributes 參數傳入額外屬性

```
@Html.EditorFor(model => model,
new { htmlAttributes = new { @class = "form-control" }, })
```

- 強化輸入驗證的前端支援
  - MinLength 與 MaxLength 終於支援前端驗證
  - AjaxOptions 的 JavaScript 回呼函式終於支援 this

```
@Ajax.ActionLink("Click me", "AjaxAction",
    new AjaxOptions { UpdateTargetId = "foo", OnBegin = "OnClick" })
<script>
function OnClick(jqXHR) {
    if ($(this).hasClass("foo")) {
        jqXHR.setRequestHeader("custom-header", "value");
    }
}
</script>
```

ASP.NET MVC 5 開發實戰

進階開發實務

多奇數位創意有限公司

技術總監 黃保翕 (Will 保哥)

部落格：http://blog.miniasp.com/

All services from your imperative.

AJAX Helper

AJAX 輔助方法

Ajax 輔助方法：超連結

All services from your imperative.

```
@Ajax.ActionLink("取得目前時間", "GetTime",
            new AjaxOptions { UpdateTargetId = "now" })

<div id="now"></div>
```

```
<a data-ajax="true" data-ajax-mode="replace" data-ajax-update="#now"
href="/Home/GetTime">取得目前時間</a>

<div id="now"></div>
```

Ajax 快取問題

All services from your imperative.

- IE 會對所有透過 AJAX 發出的 HTTP GET 要求進行快取!

```
[OutputCache(NoStore=true, Duration=0)]
public ActionResult GetTime()
{
    return Content(DateTime.Now.ToString("F"));
}
```

Ajax 輔助方法：表單

All services from your imperative.

- `@using (Html.BeginForm()) {`
- `@using (Ajax.BeginForm(`
  `new AjaxOptions { UpdateTargetId = "now" }))`

	Ajax 超連結功能	Ajax 表單功能
Ajax輔助方法	@Ajax.ActionLink	@Ajax.BeginForm
執行流程	1. 使用者點擊超連結 2. 對<a>超連結上的href屬性定義的Action網址發出HTTP要求 3. 取回內容後再將內容填入AjaxOptions 物件的UpdateTargetId屬性所指定的id元素中。	1. 使用者送出表單 2. 對<form>表單元素上的action屬性的Action網址發出HTTP要求，並將表單所有資料傳過去 3. 取回內容後再將內容填入AjaxOptions 物件的UpdateTargetId屬性所指定的id元素中。

瞭解 AjaxOptions 型別

All services from your imperative.

屬性名稱	說明
UpdateTargetId	設定回傳值要顯示在哪一個ID上。
Confirm	執行Ajax之前會先跳出一個確認對話框。
HttpMethod	設定HTTP要求的方法 (Get或Post)。
InsertionMode	設定透過Ajax輔助方法取回資料時，要如何將資料新增至UpdateTargetId指定的元素中，有三種方法： InsertionMode.Replace：取代UpdateTargetId的內容。(預設) InsertionMode.InsertBefore：在UpdateTargetId之前插入。 InsertionMode.InsertAfter：在UpdateTargetId之後插入。
LoadingElementId	在Ajax尚未完成所有工作前顯示的元素Id值。
OnBegin	設定開始時要執行的JavaScript函式名稱。
OnComplete	設定結束時要執行的JavaScript函式名稱。
OnFailure	設定失敗時要執行的JavaScript函式名稱。
OnSuccess	設定完成時要執行的JavaScript函式名稱。
Url	設定Ajax Request的網址。

整合運用

All services from your imperative.

□ 範例程式 1

```
@Ajax.ActionLink("刪除資料", "GetTime",
        new {controller="Home", id = 3 },
        new AjaxOptions {
                OnSuccess = "Delete",
                Confirm="你確定要刪除嗎？",
                HttpMethod="POST",
                LoadingElementId="ajaxLoad" })

<script>
    function Delete(data) {
        alert(data);
    }
</script>
```

整合運用

All services from your imperative.

◻ 範例程式 2 - 取用 this 物件

```
@Ajax.ActionLink("刪除資料", "GetTime",
        new {controller="Home", id = 3 },
        new AjaxOptions {
                OnSuccess = "OnClick",
                Confirm="你確定要刪除嗎？",
                HttpMethod="POST"})

<script>
    function OnClick(data) {
        if($(this).hasClass('foo')) {
            alert(data);
        }
    }
</script>
```

All services from your imperative.

Bundling and Minification

合併與壓縮 JAVASCRIPT / CSS 輸出

使用 Unobtrusive Client Validation

All services from your imperative.

- 何謂 Unobtrusive JavaScript
  - 一種將Javascript從HTML結構抽離的設計概念
- 看看 HTML / JavaScript 的差別
  - Unobtrusive Client Validation in ASP.NET MVC 3
- 啟用 Unobtrusive JavaScript 的方式
  - 設定 Web.config 或透過 C# 程式碼設定啟動
  - 一定要載入 3 個檔案
    - ~/Scripts/jquery-1.7.2.js
    - **~/Scripts/jquery.validate.js**
    - **~/Scripts/jquery.validate.unobtrusive.js**

用戶端驗證方式 – 在 Web.config 設定

All services from your imperative.

- 設定選項
  - ClientValidationEnabled
    - 啟用「用戶端驗證」意即使用 JavaScript 驗證
  - UnobtrusiveJavaScriptEnabled
    - 啟用 “Unobtrusive JavaScript” 的運作方式

```
<appSettings>
  <add key="webpages:Version" value="1.0.0.0" />
  <add key="ClientValidationEnabled" value="true" />
  <add key="UnobtrusiveJavaScriptEnabled" value="true" />
</appSettings>
```

用戶端驗證方式 – 在 Controller 指定

All services from your imperative.

- 設定方法
  - HtmlHelper.ClientValidationEnabled = true;
  - HtmlHelper.UnobtrusiveJavaScriptEnabled = true;

```
public ActionResult Create()
{
    HtmlHelper.ClientValidationEnabled = true;
    HtmlHelper.UnobtrusiveJavaScriptEnabled = true;

    return View();
}
```

快速上手

All services from your imperative.

- 全域變數
  - BundleTable.Bundles
- 預設路徑
  - \App_Start\BundleConfig.cs
- 基本語法

```
bundles.Add(new ScriptBundle("~/bundles/jquery")
    .Include("~/Scripts/jquery-1.*"));

bundles.Add(new StyleBundle("~/Content/css")
    .Include("~/Content/site.css"));

bundles.Add(new StyleBundle("~/Content/themescss")
    .IncludeDirectory("~/Content/themes/base", "*.css"));
```

在 View 裡載入的方式

All services from your imperative.

使用 @Styles 與 @Scripts 載入合併版本

```
@Styles.Render("~/Content/themes/base/css",
               "~/Content/css")

@Scripts.Render("~/bundles/jquery")
```

- 在不啟用壓縮的情況下預設會自動忽略以下 3 種檔名格式:
  - *-vsdoc.js
  - *.min.js
  - *.debug.js

啟用 JS/CSS 最佳化功能 (最小化)

All services from your imperative.

在 web.config 中

```
<compilation debug="false" targetFramework="4.5" />
```

在 BundleConfig.cs 中

```
BundleTable.EnableOptimizations = true;
```

- 註：BundleTable 是靜態類別(System.Web.Optimization)

注意：

- 在**啟用最佳化**的情況下預設會載入 *.min.js

關於 JS/CSS 的合併與輸出

All services from your imperative.

- 不用再明確指定 JS/CSS 檔名在 _Layout.cshtml 檔案裡面
- 預設會送出 HTTP Expires 標頭快取 1 年
- 合併壓縮後的輸出網址格式
  - http://xx.com/bundles/jquery?v=LsMCe96-p5oJgNHYGGB4L5iBU-9lwZIN8ggQ5ePtESY1
  - 其中 v 的值來自於所有壓縮過後的檔案內容進行雜湊運算出來的 hash 值，內容有變更才會改變！

注意事項

All services from your imperative.

- 路由命名要注意
  - 不要跟其他 Controller 衝突了！

```
bundles.Add(new ScriptBundle("~/bundles/jquery")
    .Include("~/Scripts/jquery-1.*"));

bundles.Add(new StyleBundle("~/Content/css")
    .Include("~/Content/site.css"));

bundles.Add(new StyleBundle("~/Content/themescss")
    .IncludeDirectory("~/Content/themes/base", "*.css"));
```

- 啟用最佳化時，會優先選擇 *.min.js 喔！
- 停用最佳化時，不會載入 *.min.js 喔！

解決無法載入 *.min.js 的問題

All services from your imperative.

```
  bundles.IgnoreList.Clear();
bundles.IgnoreList.Ignore("*.intellisense.js");
bundles.IgnoreList.Ignore("*-vsdoc.js");
bundles.IgnoreList.Ignore("*.debug.js", OptimizationMode.WhenEnabled);
// bundles.IgnoreList.Ignore("*.min.js", OptimizationMode.WhenDisabled);
// bundles.IgnoreList.Ignore("*.min.css", OptimizationMode.WhenDisabled);

!!! 請注意 !!!
Microsoft.AspNet.Web.Optimization.1.1.3 開始改用 bundles.DirectoryFilter 設定！

bundles.DirectoryFilter.Clear();
bundles.DirectoryFilter.Ignore("*.intellisense.js");
bundles.DirectoryFilter.Ignore("*-vsdoc.js");
bundles.DirectoryFilter.Ignore("*.debug.js", OptimizationMode.WhenEnabled);
// bundles.DirectoryFilter.Ignore("*.min.js", OptimizationMode.WhenDisabled);
// bundles.DirectoryFilter.Ignore("*.min.css", OptimizationMode.WhenDisabled);
```

All services from your imperative.

PagedList.Mvc

資料分頁的開發技巧

PagedList.Mvc 三部曲

All services from your imperative.

- 透過 NuGet 下載套件
  - (官網) Install-Package PagedList.Mvc
  - https://github.com/TroyGoode/PagedList
- 修正 Controller 套用分頁設定
  - .ToPagedList(pageNumber: 1, pageSize: 10);
- 將列表頁面套上 PagedList Helper
  - 載入樣式 PagedList.css
  - 輸出分頁 HTML 語法 @Html.PagedListPager()

實作分頁功能的三個主要工作

All services from your imperative.

- 取得 IEnumerable 或 IQueryable 型別 (或衍生型別) 的來源資料。
  - `var data = db.Product.OrderBy(p => p.ProductId);`
- 透過 ToPagedList 擴充方法，取得分頁後的結果資料，並傳入View裡。
  - `using PagedList;`
  - `var pagedData = data.ToPagedList(pageNumber: 1, pageSize: 10);`
- 在 View 頁面中透過 @Html.PagedListPager輔助方法輸出分頁導覽連結。
  - Layout
    - `<link href="~/Content/PagedList.css" rel="stylesheet" />`
  - View
    - `@using PagedList`
    - `@using PagedList.Mvc`
    - `@model IEnumerable<MvcShopping.Models.Product>`
    - `@{`
    - `    var data = Model as IPagedList;`
    - `}`
    - `@Html.PagedListPager(list: data, generatePageUrl: page => Url.Action("ProductList", new { p = page }))`

啟用 AJAX 分頁的方法

All services from your imperative.

- 在 View 套用
  - if (IsAjax) { Layout = ""; }
- Html.PagedListPager 可以傳入第三個參數 (options)
  - PagedListRenderOptions.EnableUnobtrusiveAjaxReplacing("#Main")
- 在 Layout 的 @RenderBody 之前加上 id="Main“
  - <section class="content-wrapper main-content clear-fix" id="Main">
  - @RenderBody()
  - </section>
- 使用範例

```
@Html.PagedListPager(
    list: Model as IPagedList,
    generatePageUrl: pageNo => Url.RouteUrl(new { page = pageNo }),
    options: PagedListRenderOptions.EnableUnobtrusiveAjaxReplacing("#Main"))
```

All services from your imperative.

Security in ASP.NET MVC

強化網站安全性

避免跨網站腳本攻擊(XSS)

All services from your imperative.

- 編碼方法
  - Html.Encode
  - Url.Encode
  - HttpUtility.UrlPathEncode
  - Ajax.JavaScriptStringEncode
  - HttpUtility.JavaScriptStringEncode

```
<script type="text/javascript">
    var lastError = '@Ajax.JavaScriptStringEncode(ViewBag.LastError)';
    alert(lastError);
</script>
```

強化表單安全性

All services from your imperative.

- 使用 AntiForgeryToken 輔助方法
  - [ValidateAntiForgeryToken]
- View 的用法
  - 表單必須要以Html.BeginForm()宣告
  - @Html.AntiForgeryToken()

Index.cshtml | Login.cshtml | MemberController.cs

```
1   @model MvcShopping.Models.MemberLoginViewModel
2
3   <h2>會員登入</h2>
4
5   @using (Html.BeginForm()) {
6
7       @Html.AntiForgeryToken()
8
9       @Html.ValidationSummary(true)
10
11      <fieldset>
12          <legend>請輸入您的帳號、密碼</legend>
```

All services from your imperative.

ASP.NET Identity

ASP.NET IDENTITY 身分驗證機制

學習資源

All services from your imperative.

- ASP.NET Identity 教學影片
  - ASP NET MVC 5 新功能探索之《ASP.NET Identity》
    http://youtu.be/Hhy0ePBADfw
- ASP.NET Identity 簡報檔
  - ASP.NET MVC 5 新功能探索 - SlideShare
    http://www.slideshare.net/WillHuangTW/aspnet-mvc-5-32611052
- 範例程式
  - Controllers\AccountController.cs

All services from your imperative.

Understanding ASP.NET Web API

認識 ASP.NET WEB API

認識 ASP.NET Web API

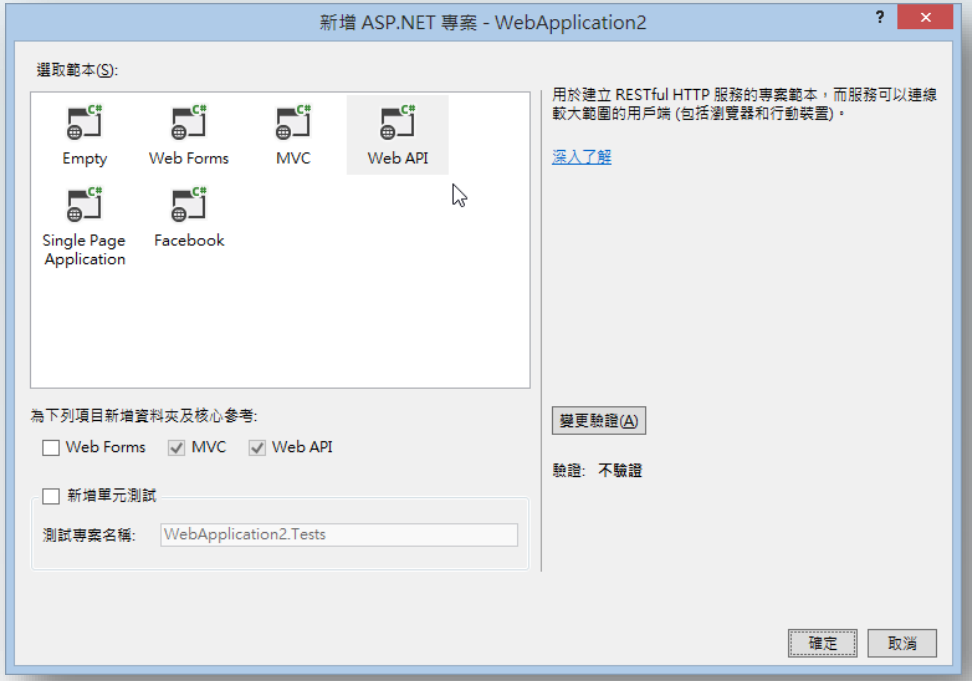
All services from your imperative.

- 起源
  - 原本的名字叫 WCF Web API，目前重新整合進 ASP.NET 產品，改名為 ASP.NET Web API
- 特性
  - 利用標準 HTTP 完成所有 API 要求
    - GET, POST, DELETE, PUT, OPTIONS, PATCH
  - 內建 XML 與 JSON 兩種回應格式
  - 支援如 ASP.NET MVC 的模型繫結特性
  - 支援 ASP.NET MVC 的 Routing 架構
  - 可使用 Action Filter

建立第一個 ASP.NET Web API 程式

All services from your imperative.

- Controller
  - ApiController
- Routing
  - MapHttpRoute
  - 僅需 {controller} 路由值



回傳型別

All services from your imperative.

- void
- 基礎型別、複雜型別
- IEnumerable<string>
  - 不支援 OData 查詢
- IQueryable<string>
  - 繼承自 IEnumerable
  - 支援 OData 查詢
    http://www.asp.net/web-api/overview/odata-support-in-aspnet-web-api
  - 適合用 Entity Framework 讀取查詢資料
- HttpResponseMessage

回傳 HttpResponseMessage 的用途

All services from your imperative.

- 用來進一步控制回應的格式與 HTTP 標頭

```
public Product PostProduct(Product item)
{
    item = repository.Add(item);

    return item;
}
```

```
public HttpResponseMessage PostProduct(Product item)
{
    item = repository.Add(item);

    var response = Request.CreateResponse<Product>(HttpStatusCode.Created, item);

    string uri = Url.Link("DefaultApi", new { id = item.Id });
    response.Headers.Location = new Uri(uri);

    return response;
}
```

All services from your imperative.

ASP.NET Web API Tips & Tricks

ASP.NET WEB API 小技巧

強迫回應 JSON 格式 (1)

All services from your imperative.

- Global.asax.cs
  - GlobalConfiguration.Configuration.Formatters.XmlFormatter.SupportedMediaTypes.Clear();

- http://blog.miniasp.com/post/2012/10/12/ASPNET-Web-API-Force-return-JSON-format-instead-of-XML-for-Google-Chrome-Firefox-Safari.aspx

強迫回應 JSON 格式 (2)

All services from your imperative.

- /api/values?output=xml
- /api/values?output=json

```
// GET api/values
public HttpResponseMessage Get(string output = null)
{
    var data = new string[] { "value1", "value2" };

    var resp = new HttpResponseMessage(HttpStatusCode.OK);

    if (output == "xml")
    {
        resp.Content = new ObjectContent<IEnumerable<string>>(data, new XmlMediaTypeFormatter());
    }
    else
    {
        resp.Content = new ObjectContent<IEnumerable<string>>(data, new JsonMediaTypeFormatter());
    }

    return resp;
}
```

Web API 例外處理方法

All services from your imperative.

```
using System.Web.Http.Filters;

public class ElmahErrorAttribute : ExceptionFilterAttribute
{
    public override void OnException(HttpActionExecutedContext actionExecutedContext)
    {
        if (actionExecutedContext.Exception != null)
            Elmah.ErrorSignal.FromCurrentContext().Raise(actionExecutedContext.Exception);

        base.OnException(actionExecutedContext);
    }
}

public static class WebApiConfig
{
    public static void Register(HttpConfiguration config)
    {
        config.Filters.Add(new ElmahErrorAttribute());

        config.Routes.MapHttpRoute(
            name: "DefaultApi",
            routeTemplate: "api/{controller}/{id}",
            defaults: new { id = RouteParameter.Optional }
        );
    }
}
```

如何在 ASP.NET Web API 套用 ELMAH 錯誤紀錄模組

無法序列化為 JSON 的狀況

All services from your imperative.

- ASP.NET Web API 無法輸出 Entity Framework 物件的解法
  http://bit.ly/WebAPI-EF-Reference-Loop
- 如何讓 ASP.NET Web API 無論任何要求都回應 JSON 格式
  http://bit.ly/Web-API-Force-JSON
- 在 Model 的屬性 (Property) 上
  - 套用 [JsonIgnore] 屬性 (Attribute)
- 在 Entity Framework 執行之前設定
  - db.Configuration.LazyLoadingEnabled = false;

All services from your imperative.

Customize T4 for ASP.NET MVC

使用 T4 程式碼範本快速開發

如何修改內建的程式碼範本 (VS2012)

All services from your imperative.

- Visual Studio 2012 自訂方法
  - 專案目錄
    - CodeTemplates
  - 預設範本路徑
    - Visual Studio 2012 在 64 位元 (x64) 下
      - **C:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\IDE\ItemTemplates\CSharp\Web\MVC 4\CodeTemplates**
    - Visual Studio 2012 在 32 位元 (x86) 下
      - C:\Program Files\Microsoft Visual Studio 11.0\Common7\IDE\ItemTemplates\CSharp\Web\MVC 4\CodeTemplates

Visual Studio 2012 自訂步驟

All services from your imperative.

- 建立 CodeTemplates 目錄
  - 拖曳 AddController 與 AddView 目錄進來
- 清空【自訂工具】屬性
  - **方案總管的專案項目屬性**
- 新增或修改 T4 範本
  - ApiController.tt
  - ApiControllerWithContext.tt
  - Controller.tt
  - ControllerWithContext.tt

其他提醒事項

All services from your imperative.

- CodeTemplates 擴充性
  - AddView 可以任意擴充
  - AddController 無法任意擴充
- ScaffoldColumn(false)
  - 不能套在 ModelMetadata
- 可考慮套用 AdditionalMetadata 屬性
  - [AdditionalMetadata(name, value)]

如何修改內建的程式碼範本 (VS2013)

All services from your imperative.

- Visual Studio 2013 自訂方法
  - 專案目錄
    - CodeTemplates
  - 預設範本路徑
    - Visual Studio 2013 在 64 位元 (x64) 下
      - C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\IDE\Extensions\Microsoft\Web\Mvc\Scaffolding\Templates
  - 參考資料
    - Modifying the default code generation/scaffolding templates in ASP.NET MVC
      http://bit.ly/VS2013-Scaffold

如何修改內建的程式碼範本 (VS2015)

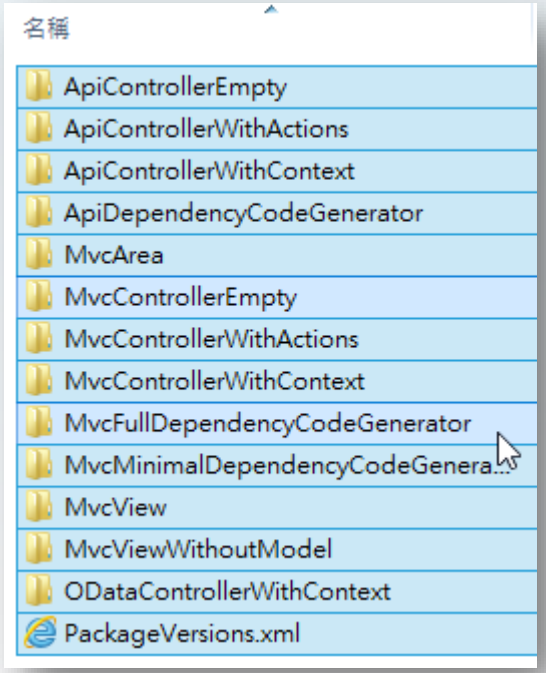
All services from your imperative.

- Visual Studio 2015 自訂方法
  - 專案目錄
    - CodeTemplates
  - 預設範本路徑
    - Visual Studio 2013 在 64 位元 (x64) 下
      - C:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\IDE\Extensions\Microsoft\Web\Mvc\Scaffolding\Templates
  - 參考資料
    - Modifying the default code generation/scaffolding templates in ASP.NET MVC
      http://bit.ly/VS2013-Scaffold

Visual Studio 2013/2015 自訂步驟

All services from your imperative.

- 建立 CodeTemplates 目錄
  - 拖曳【預設範本路徑】所有目錄進來
- 清空【自訂工具】屬性
  - **方案總管的專案項目屬性**
- 新增或修改 T4 範本



聯絡資訊

All services from your imperative.

- The Will Will Web

  記載著 Will 在網路世界的學習心得與技術分享

  - http://blog.miniasp.com/

- Will 保哥的技術交流中心 (臉書粉絲專頁)

  - http://www.facebook.com/will.fans

- Will 保哥的噗浪

  - http://www.plurk.com/willh/invite

- Will 保哥的推特

  - https://twitter.com/Will_Huang