以實例說明 ASP.NET Web API 服務的開發與測試過程

Kevin Tseng



簡介

- 微軟最有價值專家（MVP）2013~2015
- twMVC 核心成員及講師 （http://mvc.tw）
- SkillTree 專任講師 （http://skilltree.my）
- 部落格「mrkt 的程式學習筆記」
  http://kevintsengtw.blogspot.tw
- Google+ 專頁
  https://plus.google.com/+KevintsengtwBlogspot
- Github
  https://github.com/kevintsengtw

主題說明

現在越來越多服務都使用 ASP.NET Web API 建立，從開發前的規劃、開發進行實作，一直到服務上線，這些過程應該要做些什麼與注意什麼，在開發的同時如何導入單元測試以及如何實作開發完成後的整合測試。

以一個已上線 APP 後端的 ASP.NET Web API 服務做為實際案例來說明，以及開發的心路歷程。

Agenda

- 好房網買屋App簡介
- 專案架構
- 開發規範
- 開發工具與套件
- 監控與紀錄
- 單元測試與整合測試
- 開發心得與感想

Agenda

- 開發時的相關參考書籍和學習資源
- 補充資料

好房網買屋App簡介

好房網買屋



http://buy.housefun.com.tw/

好房網買屋

- .NET Framework 4.0
- VB.NET
- ASP.NET WebForm, Web Site Project(網站專案)
- ADO.NET
- Microsoft SQL Server
- Visual Studio 2012
- TFS(單一鎖定)

好房網買屋 APP



http://apps.housefun.com.tw/apps/buy/

好房網買屋 APP

- .NET Framework 4.5
- C# 5.0
- ASP.NET Web Api 2.2.3
- ADO.NET, Dapper
- Microsoft SQL Server
- Visual Studio 2013
- Git

開發小事記

- 2014-10-01 專案啟動（規格制訂、架構設計、專案試做）
- 2014-11-06 開始進入開發實作
- 2014-12-25 加入測試專案
- 2014-12-29 加入使用 Quatrz.NET
- 2015-01-09 加入使用 NanoProfiler
- 2015-01-15 資料存取改用 Dapper
- 2015-02-24 測試專案加入使用 Fluent Assertions

開發小事記

- 2015-03-13 發佈第一個正式版本
- 2015-05-18 測試專案加入使用 ExpectedObjects
- 2015-06-05 建立獨立專案以隔離公司底層元件
- 2015-08-07 進行會員相關功能的開發實作
- 2015-09-03 完成會員相關功能並開始進行整合測試
- 2015-10-19 Help Page, Swagger, 匯入 Postman 功能
- 2015-11-20 第一階段功能開發完成(結束這一回合)

APP 大事記

- 2015-05-13 Android Play 上架
- 2015-07-07 iPhone App Store 上架（雙版本上線）
- 2015-07-24 Apple App Store 編輯精選
  最佳新 App 第一位
- 2015-09-09 第13屆金企鵝獎 最佳平台產品獎
- 2015-11-05 改版上線，加入會員功能

開發前面臨到什麼狀況？

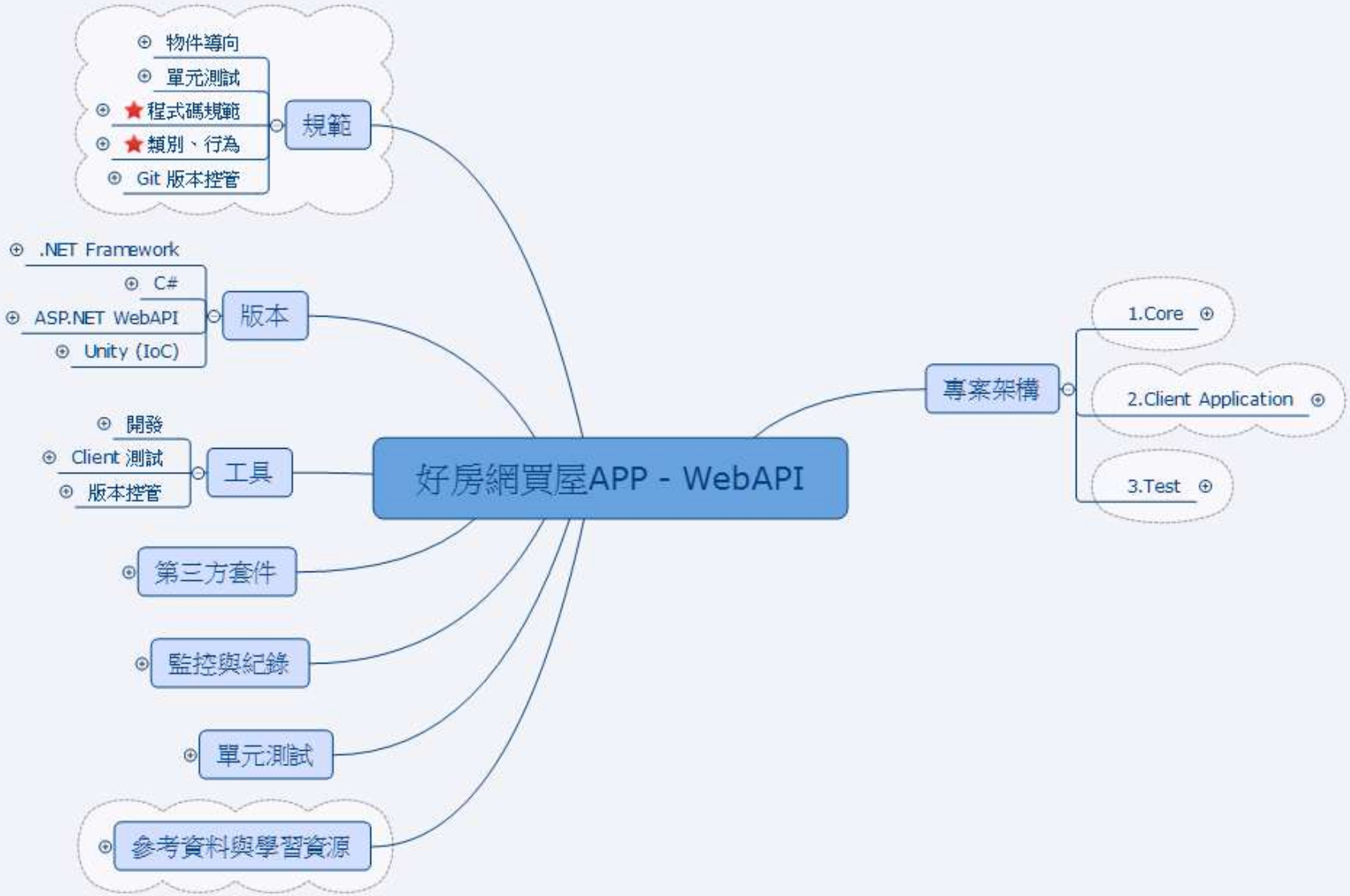
- Domain Know-How
- 專案結構的不同
- 使用技術的差異
- 商業邏輯與資料存取處理混雜與四散
- 既有系統的開發並非使用物件導向設計
- 資料庫的 Table 沒有正規化
- 系統設計與功能規劃並沒有詳盡的統一處理與規範

有什麼樣的挑戰？

- 從既有專案與底層元件裡去找出相關與能用的程式
- 從相關與能用的程式中分離出資料存取和商業邏輯
- 公司底層元件
- 制訂一致性的輸出入處理與規範
- 效能問題

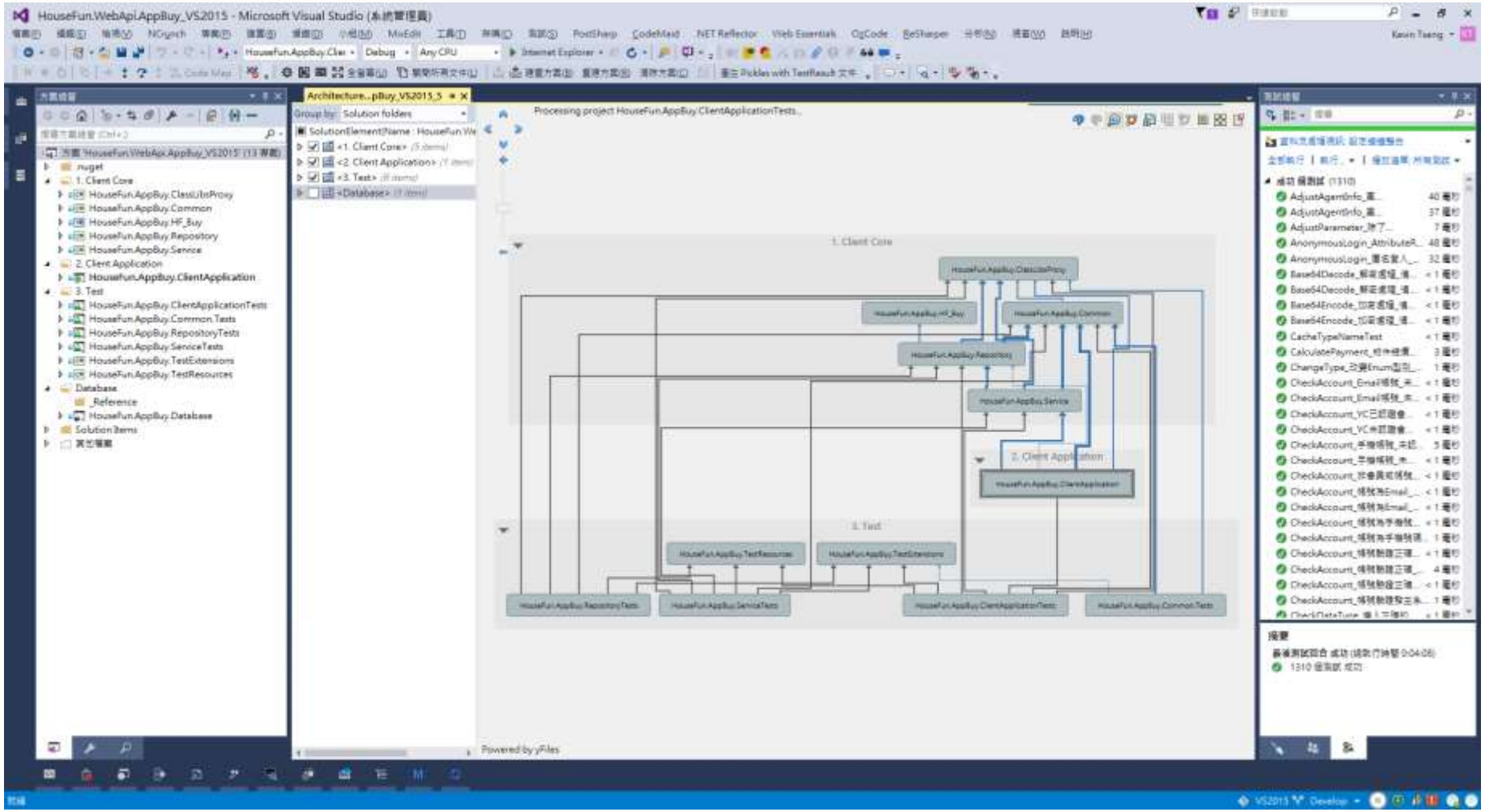
有什麼樣的挑戰？

- 這個專案是給公司之後新的 Web Api 專案的開發範本
- 開發期間除了這個專案，我還必須參與其他專案
- 教育訓練
- 兼任公司內的非正式的技術支援
- 兼任 QA
- ***原本這個專案是要我帶著一個小團隊做開發，***
  但最後莫名其妙地就被我一個人吃了下來

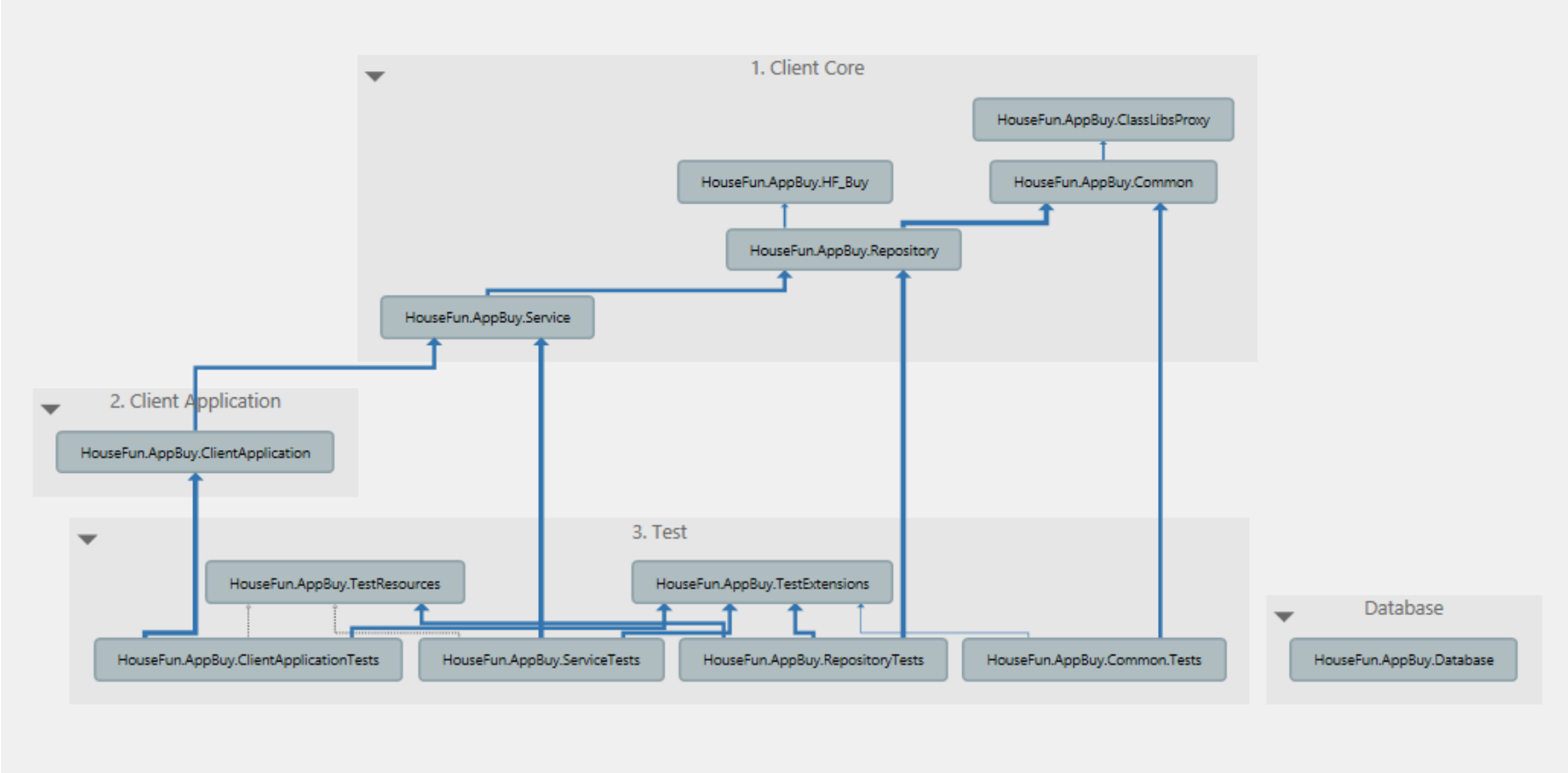

好房網買屋APP - WebAPI
規範
物件導向
單元測試
程式碼規範
類別、行為
Git 版本控管
版本
.NET Framework
C#
ASP.NET WebAPI
Unity (IoC)
工具
開發
Client 測試
版本控管
第三方套件
監控與紀錄
單元測試
參考資料與學習資源
專案架構
1.Core
2.Client Application
3.Test

專案架構

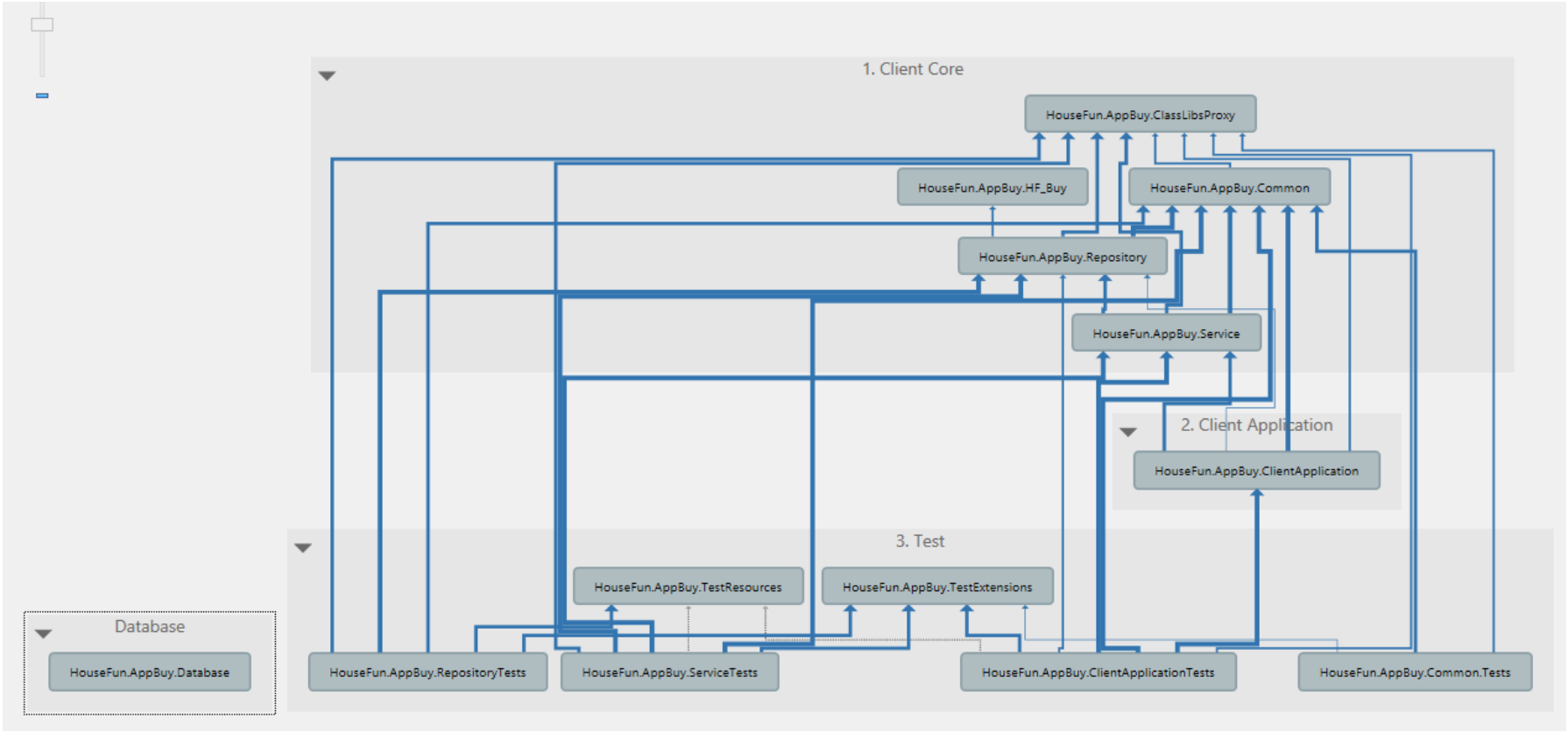
好房網買屋 APP Web API 專案



好房網買屋 APP Web API 專案



好房網買屋 APP Web API 專案



好房網買屋 APP Web API 專案

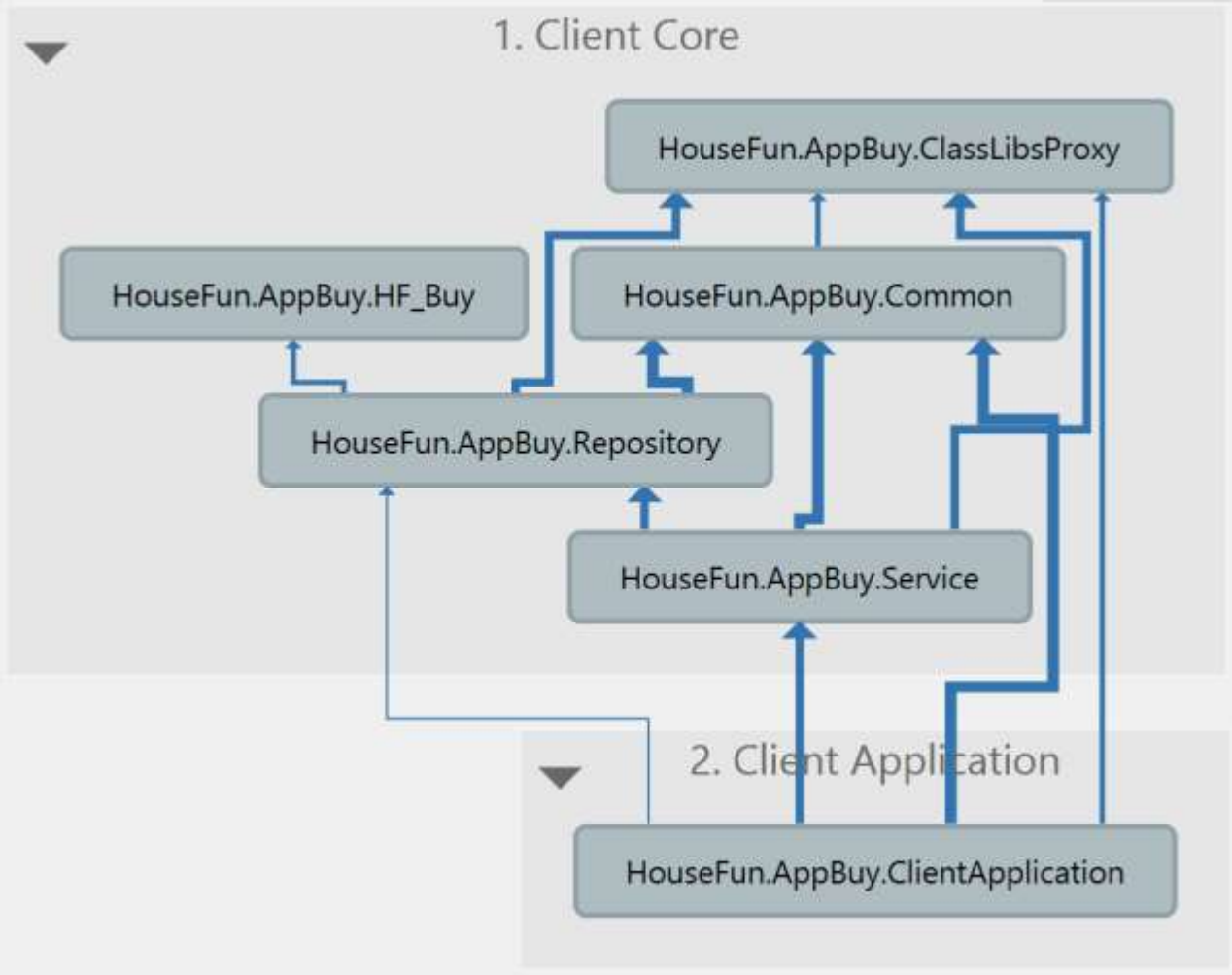
- Solution 架構分層規劃區分為：

  Repository (資料存取)

  Service (商業邏輯)

  ClientApplication (Web API 專案)

好房網買屋 APP Web API 專案



好房網買屋 APP Web API 專案

- 各專案之間不直接建立服務類別的實體，而是使用介面，所以會使用到 IoC Container
- 這個專案用的是：

  Microsoft Pattern&Practices Unity

公司底層元件的隔離

- **專案內不允許直接使用既有的公司底層元件**
- 會使用到的公司底層元件必須另外建立代理，

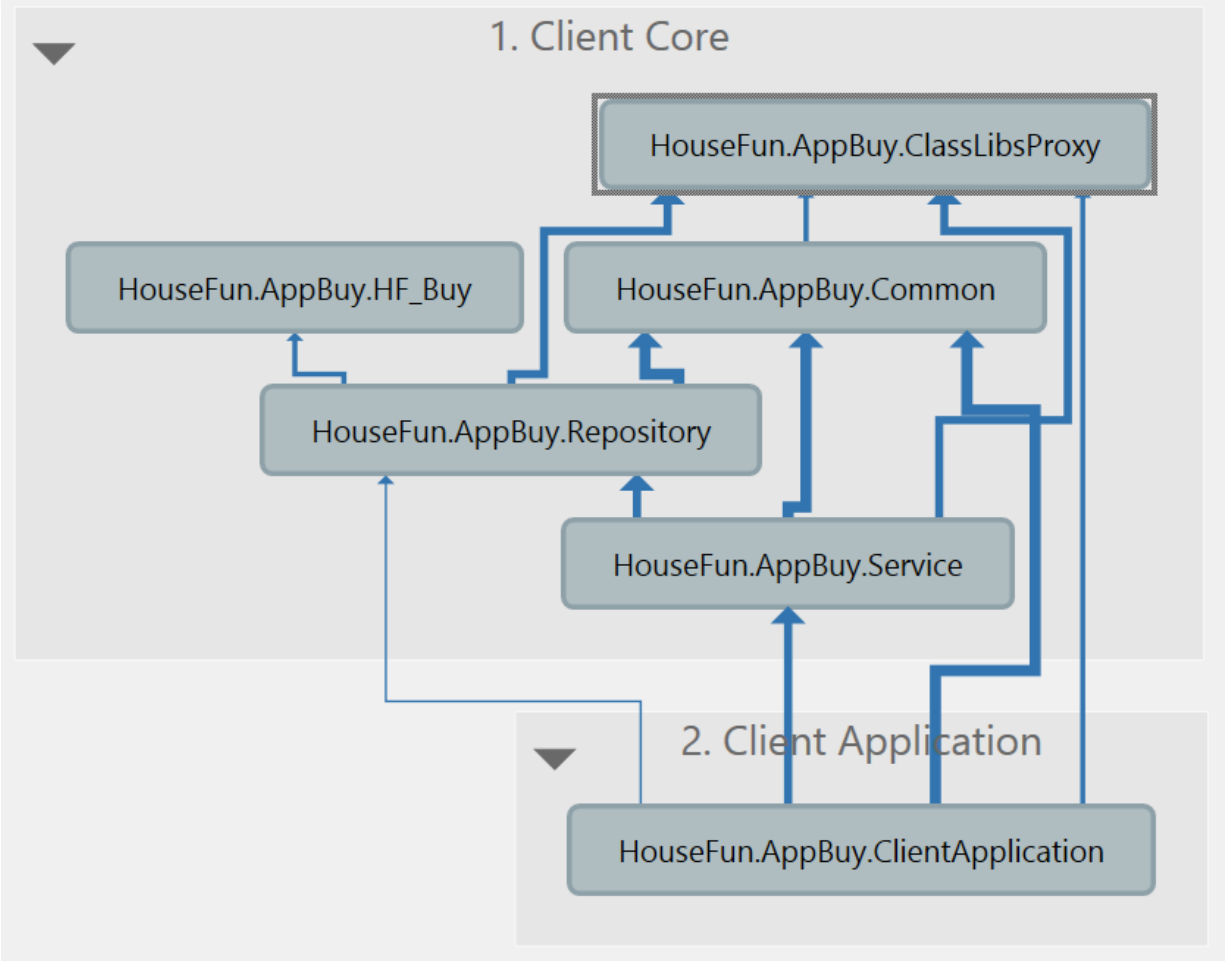
  專案呼叫使用代理類別，

  代理類別再去呼叫使用原有的底層元件

- **不允許直接使用的原因在於，不隔離就無法測試**

  因為公司底層元件太多直接相依於外部資源

各專案透過代理去操作公司底層元件



開發規範

物件導向

- 物件導向程式設計 OOP
- 物件導向程式設計的基本準則 – SOLID

  SRP – Single Responsibility Principle

  OCP – Open-Close Principle

  LSP – Liskov's Substitution Principle

  LKV – Least Knowledge Principle

  ISP – Interface Segregation Principle

  DIP – Dependency Inversion Principle

設計模式？

- 只強調必須遵循「SOLID 物件導向設計原則」

  不強調使用「Design Pattern 設計模式」

- 設計模式是用來解決問題，而不是用來做為設計準則

- 有使用：

  獨體模式、工廠模式、代理模式… etc

  其他也有用到，但並不是為了要用模式而用，

  而是為了要解決問題而用

單元測試

- 主要流程一定要做到單元測試
- 測試沒有通過，不可以將程式碼簽入版控
- 凡是修改而造成原有測試不通過，就有責任將影響的單元測試都修改為可通過
- 資料層盡量不要直接連接開發或測試資料庫，連接資料庫的測試，須使用 LocalDB

程式碼規範

- 服務層、資料層、應用層之間不可直接相依，

  一律透過 Interface 並使用 IoC Container 相依注入實體

- 程式碼的最大寬度不可超過 120 Columns，善用折行處理

- 變數與命名：

  程式內的變數在有用到的地方宣告與建立就可以

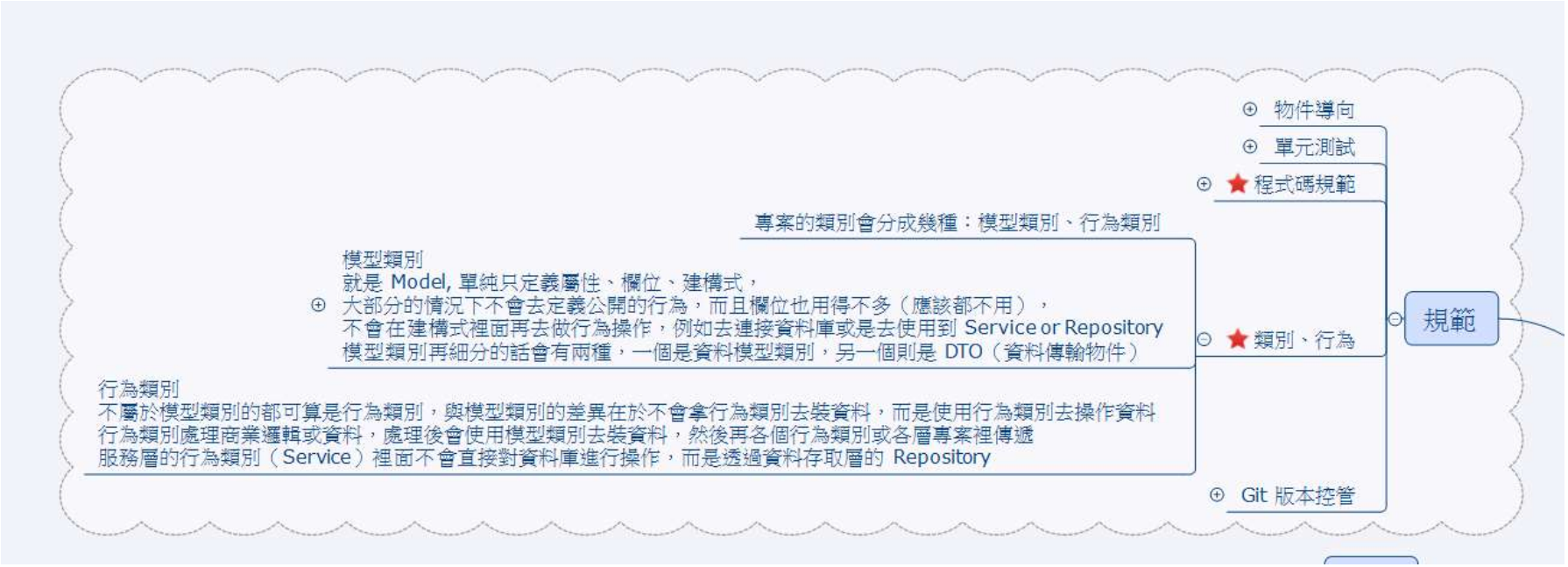
  公開的變數，請使用 Pascal 命名方式，字首為大寫

  私有的變數，可前綴底線或是字首使用小寫

  方法簽章裡的變數，一律使用 Camel Case

  私有方法，一律使用 Camel Case

模型類別、行為類別

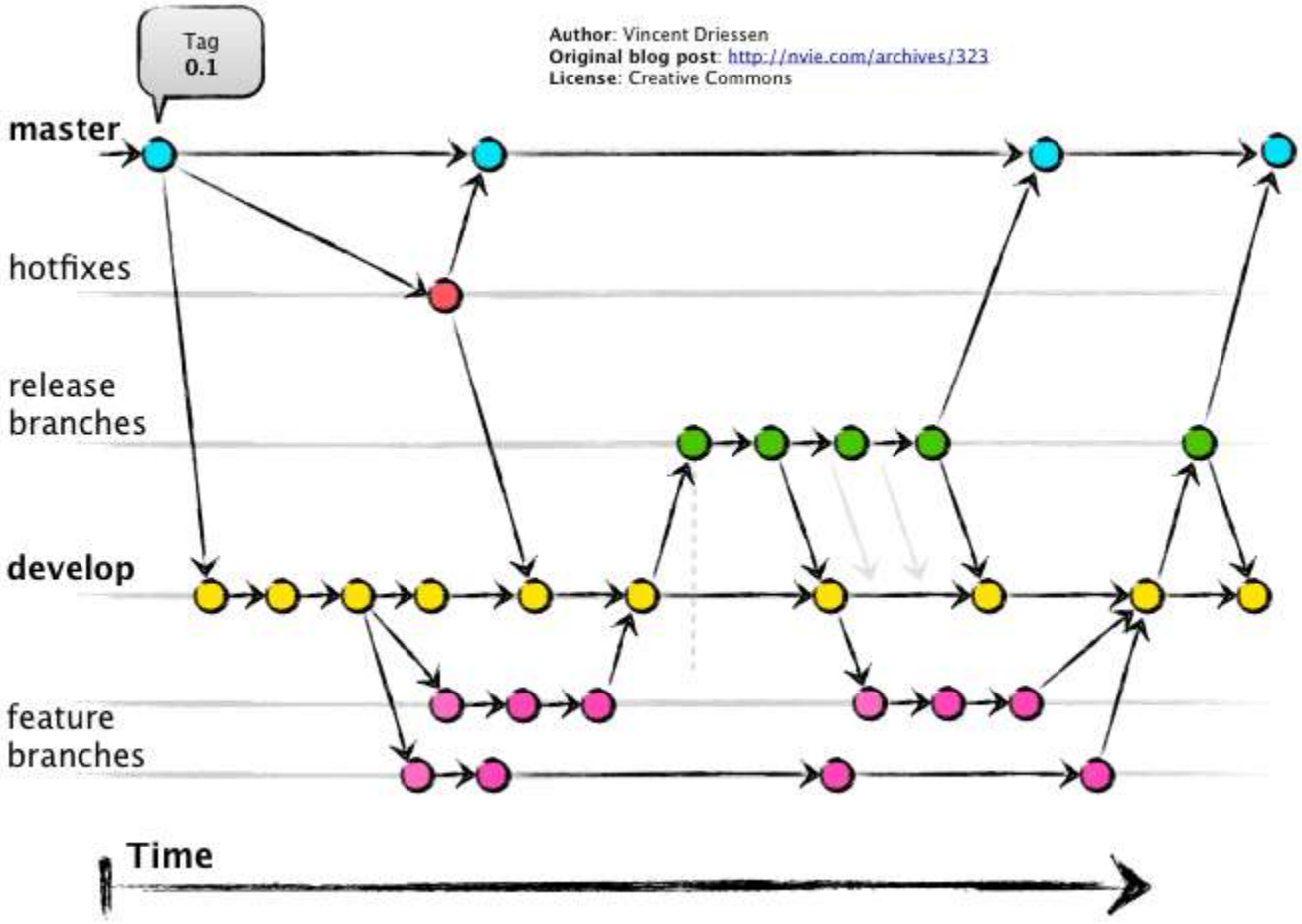

規範
物件導向
單元測試
程式碼規範
類別、行為
專案的類別會分成幾種：模型類別、行為類別
模型類別
就是 Model, 單純只定義屬性、欄位、建構式，
大部分的情況下不會去定義公開的行為，而且欄位也用得不多（應該都不用），
不會在建構式裡面再去做行為操作，例如去連接資料庫或是去使用到 Service or Repository
模型類別再細分的話會有兩種，一個是資料模型類別，另一個則是 DTO（資料傳輸物件）
行為類別
不屬於模型類別的都可算是行為類別，與模型類別的差異在於不會拿行為類別去裝資料，而是使用行為類別去操作資料
行為類別處理商業邏輯或資料，處理後會使用模型類別去裝資料，然後再各個行為類別或各層專案裡傳遞
服務層的行為類別（Service）裡面不會直接對資料庫進行操作，而是透過資料存取層的 Repository
Git 版本控管

DTO, Model 類別

- 各專案間的資料傳遞會透過 DTO 物件，很少會有一個資料型別從頭用到尾的狀況
- 大部分的情況是兩個 Layer 之間會有共同的 DTO 類別
- 資料庫存取的對應類別為 Repository 的 Model 類別
- Service 的對應類別為 Service 專案內的 DTO 類別
- Web Api 的類別有：

  ParameterModel, ParameterDto, ViewModel

Git 版本控管 - GitFlow



開發規範


規範
物件導向
SOLID
SRP Single Responsibility Principle
OCP Open-Closed Principle
LSP Liskov's Substitution Principle
ISP Interface Segregation Principle
DIP The Dependency Inversion Principle
單元測試
主要流程一定要做到單元測試
測試沒有通過，不可以將程式碼簽入版控
凡是修改而造成原有測試不通過，就有責任將影響的單元測試都修改為可通過
資料層盡量不要直接連接使用 DEV 或 DIFF 資料庫，要使用 LocalDB
服務層的測試不可以直接連接資料庫，如需要就須使用 LocaalDB
程式碼規範
類別、行為
Git 版本控管
主要發行的 Branch 為 master
不可以在 master 裡面去做任何程式修改
開發的主要分支為 Develop
各分支的開發最後要合併回 Develope 這條線
各開發分支的簽入要持續與頻繁
不要一天才簽入一個版本
只要功能開發到一個階段，可完成編譯以及通過所有的測試後，就 Commit (簽入) 與 Sync (同步處理)
簽入時物必要將該次簽入的內容在備註裡寫清楚
之後在找的時候才會比較清楚（而且對寫工時系統會有幫助）

開發工具與套件

開發工具

- Visual Studio 2013 Ultimate
- LINQPad Premium
- Microsoft SQL Server Management Studio
- Postman

主要的 Visual Studio 擴充功能

- Telerik JustCode
- OzCode
- NCrunch
- Unit Test Generator
- CodeMaid
- SmartPaster 2013
- Microsoft CodeLens Code Health Indicator

專案使用的第三方套件

- AutoMapper
- CsvHelper
- Dapper
- GeoJSON.Net
- Microsoft Pattern&Practices Unity 3.5

  Unity.AspNet.WebApi
- Newtonsoft.Json (JSON.Net)

特別介紹以下的套件和工具

- 套件：

  Dapper

  ASP.NET Web Api Help Page

  Swashbuckle (Swagger for Web Api)

- 工具：

  Microsoft CodeLens Code Health Indicator

  Postman

Repository – 為何不用 ORM?

使用 ORM 是必須從專案設計的一開始就要去做整體的規劃

除了專案裡的類別、程式之外，

也必須包含資料庫和資料表本身

Repository – 為何不用 ORM?

既有的資料表(Table)存在著太多太多的問題，

Table 本身就不是依照物件去做設計，

而且單一個 Table 並不是只有你在使用，

多個專案與開發團隊都會去踢一腳、摸一把，

以這樣為基礎所做的 ORM Model 會讓開發者天天撞牆



Repository – 為何不用 ORM?

- 你有看過一個欄位會在不同的專案、不同的時期就會有不同的定義與用法嗎？
- 你有看過一個數值欄位會因為正負值有不同用法、低於某個值的時候還有不同的解釋？
- 更多是因為欄位太多、歷史過於悠久，但是因為不知道有哪些專案有再用，所以不能移除也不敢變動



專案為什麼不使用 DataSet, DataTable ?

- 領略過 ORM (EF,LINQ to SQL) 在開發上的好處，
  就不會想再回到操作 DataSet,DataTable… 的回頭路
- ~~當你去過了天堂，你還會想回到地獄嗎？~~
- ~~LINQ or Die !~~



對映到類別？為何不自行處理對映？

直經用 ADO.NET 執行 SQL Command 後的回傳結果，

不是也可以自己去處理資料與欄位的對映嗎？

- 當欄位多而且每一筆資料欄位幾乎都是數十甚至上百個時，你就不會想要這麼做
- 更千萬不要想自己去用 Reflection 處理資料對映，因為效率問題就會讓你滿頭包

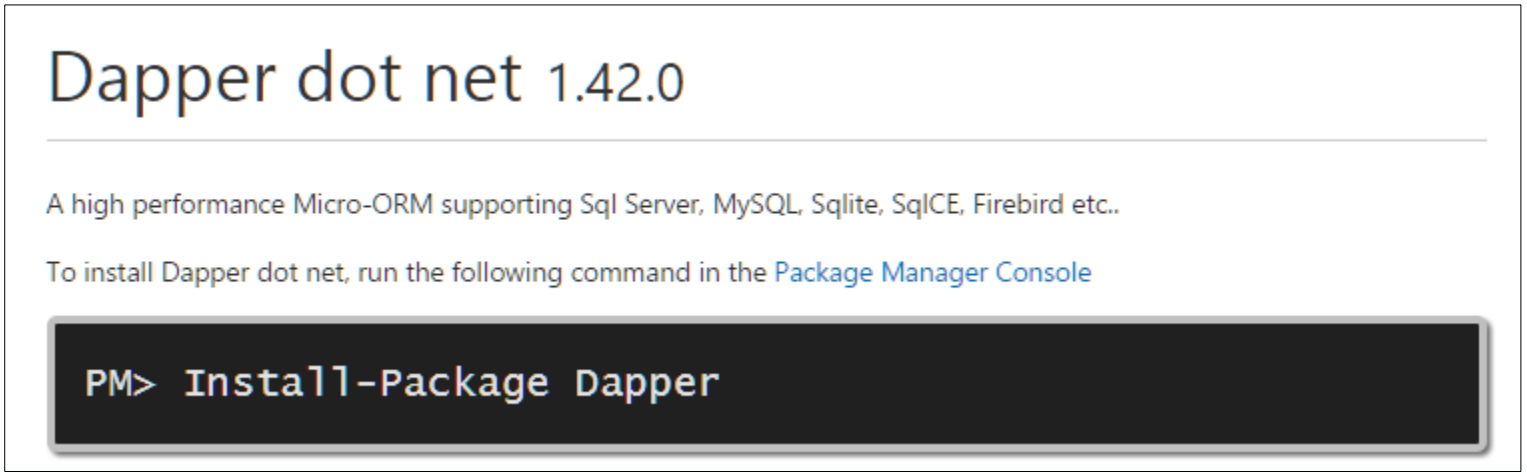
使用 Dapper

Dapper-dot-net (StackExchange)

https://github.com/StackExchange/dapper-dot-net

Data + Mapper

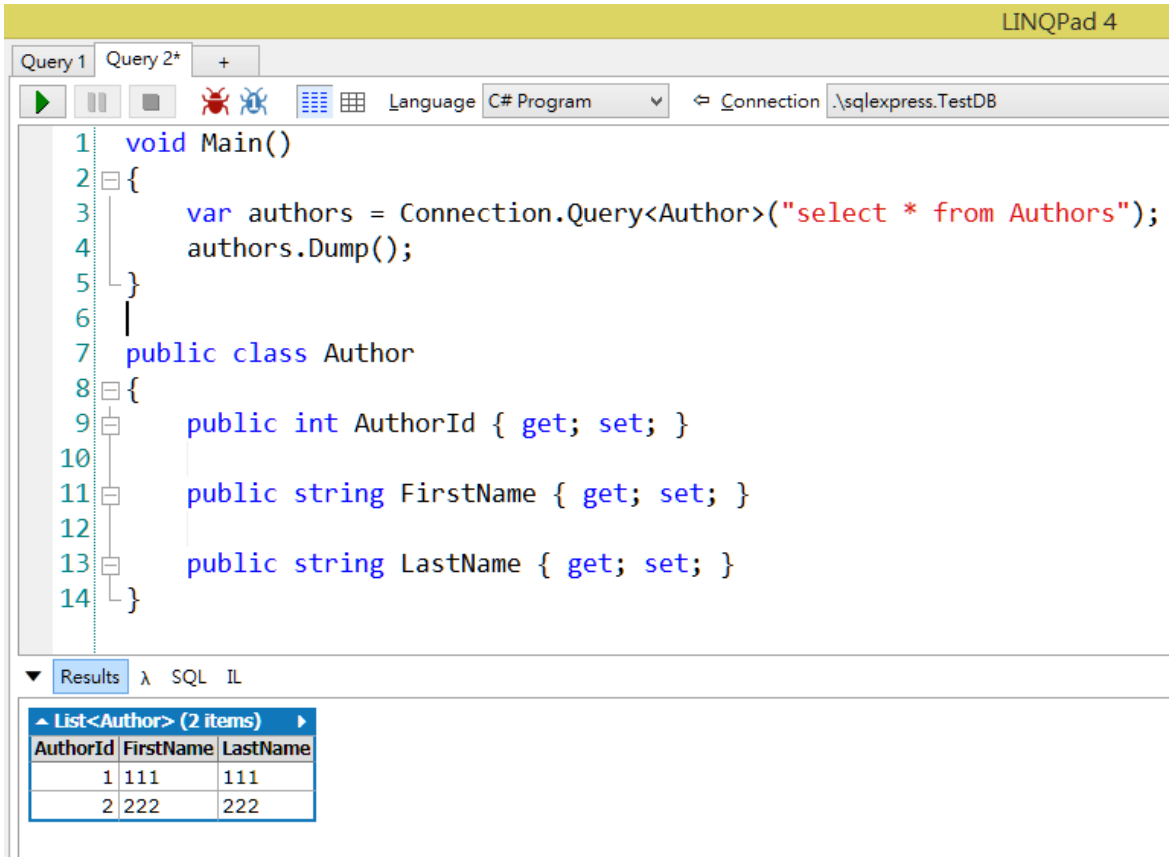
High Performance



https://www.nuget.org/packages/Dapper/

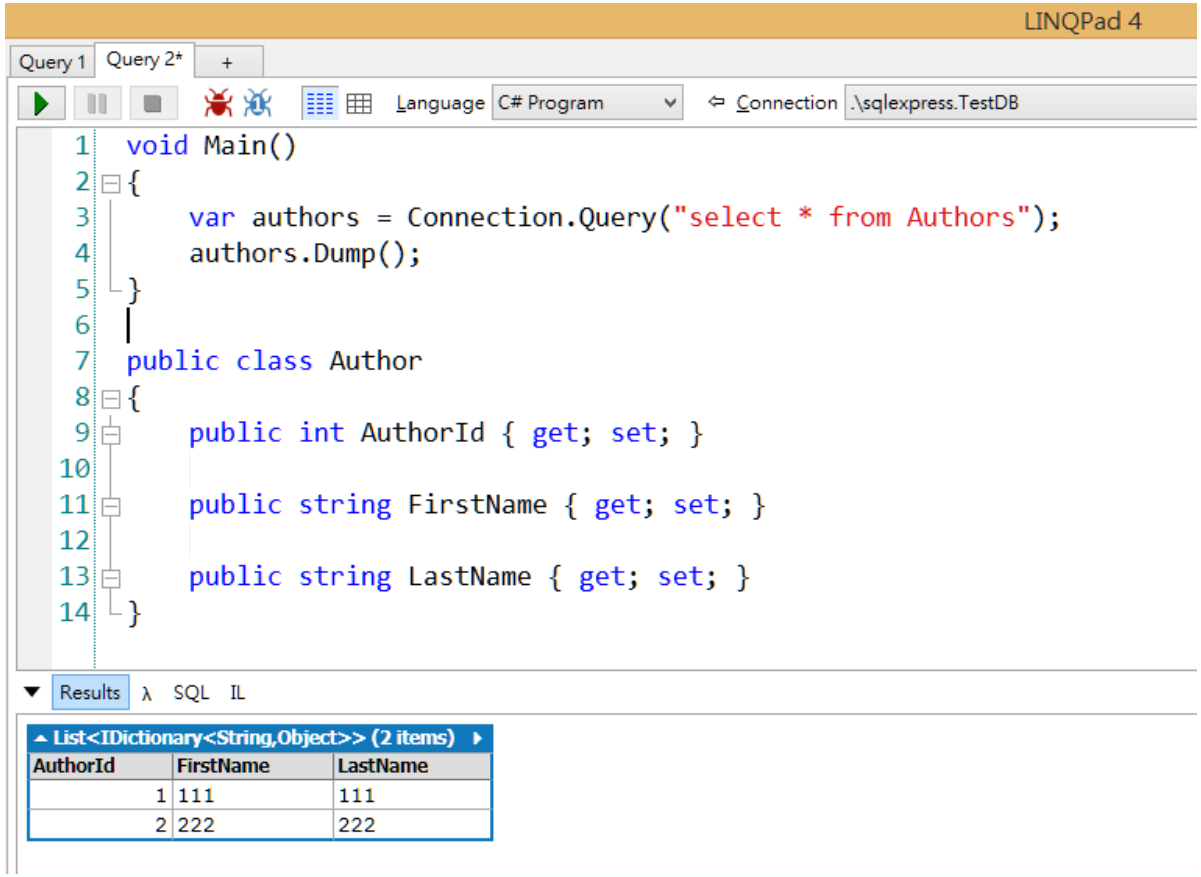
Dapper

- Execute a query and map the results to a strongly typed List



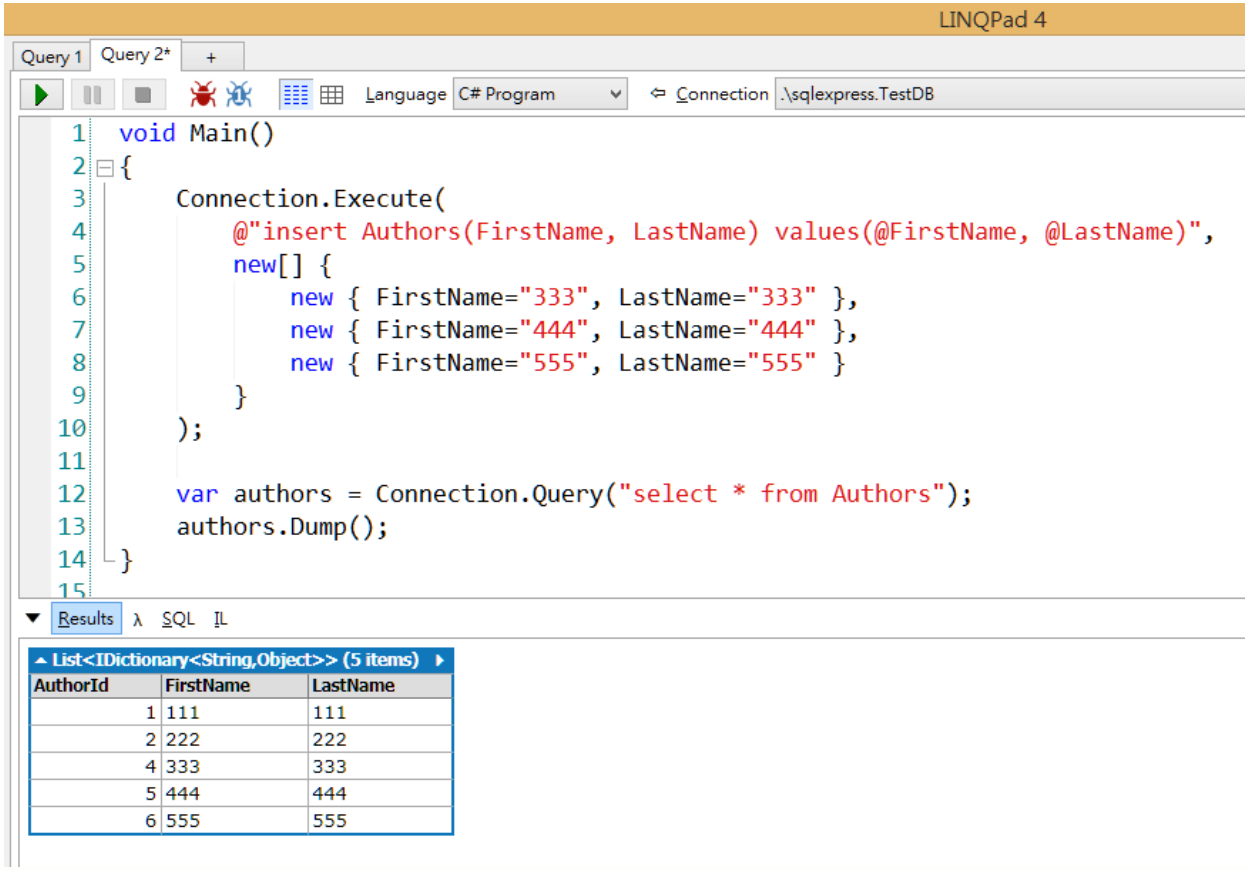
Dapper

- Execute a query and map it to a list of dynamic objects



Dapper

- Execute a Command multiple times

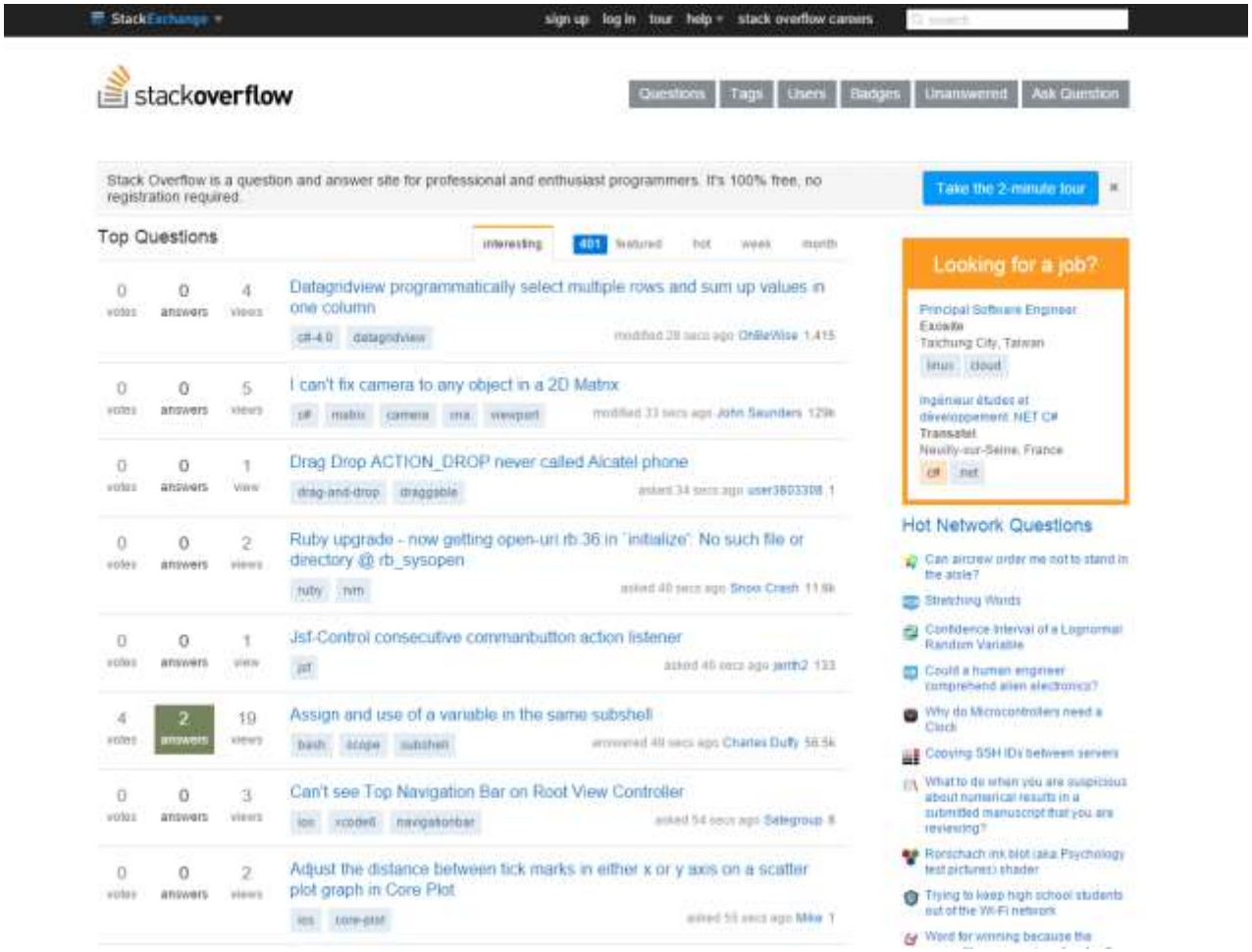


Dapper

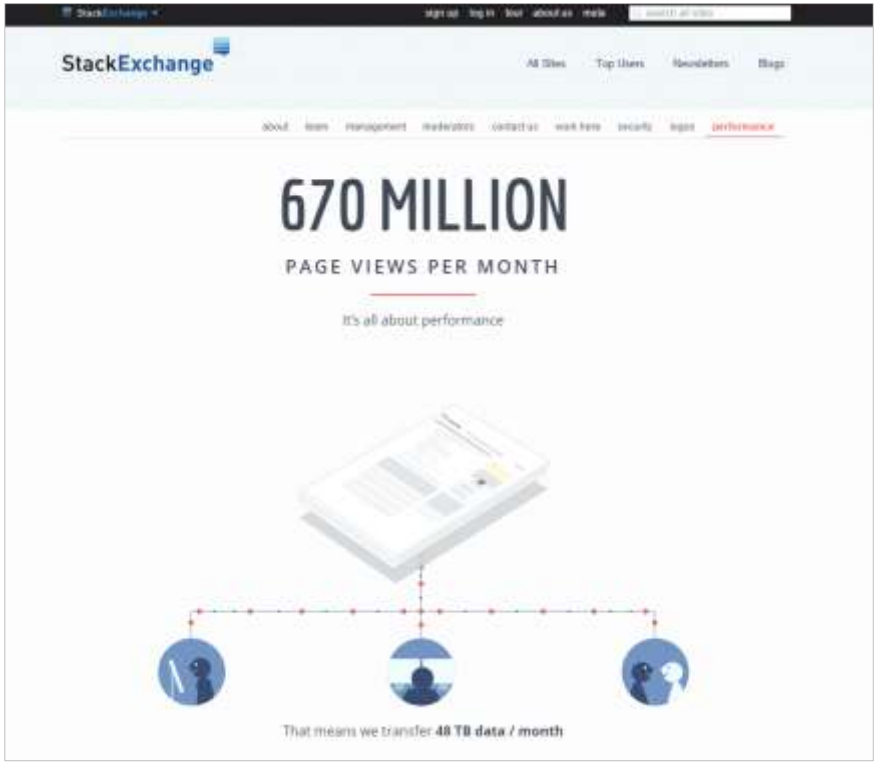
Performance of SELECT mapping over 500 iterations - POCO serialization

Method	Duration	Remarks
Hand coded (using a `SqlDataReader`)	47ms	
Dapper `ExecuteMapperQuery`	49ms	
ServiceStack.OrmLite (QueryById)	50ms	
PetaPoco	52ms	
BLToolkit	80ms	Can be faster
SubSonic CodingHorror	107ms	
NHibernate SQL	104ms	
Linq 2 SQL `ExecuteQuery`	181ms	
Entity framework `ExecuteStoreQuery`	631ms	

有誰在用 Dapper ？


StackExchange
sign up log in tour help stack overflow careers
stackoverflow
Questions Tags Users Badges Unanswered Ask Question
Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.
Take the 2-minute tour
Top Questions
interesting 401 featured hot week month
Datagridview programmatically select multiple rows and sum up values in one column
I can't fix camera to any object in a 2D Matrix
Drag Drop ACTION_DROP never called Alcatel phone
Ruby upgrade - now getting open-uri rb:36 in `initialize': No such file or directory @ rb_sysopen
Jsf-Control consecutive commanbutton action listener
Assign and use of a variable in the same subshell
Can't see Top Navigation Bar on Root View Controller
Adjust the distance between tick marks in either x or y axis on a scatter plot graph in Core Plot
Looking for a job?
Hot Network Questions

有誰在用 Dapper ？





http://stackexchange.com/performance

為何使用 Dapper ？

- 底層元件與既有系統都是透過公司自行再包裝的資料存取類別對資料庫作存取，回傳結果為 DataSet or DataTable
- 資料轉換，從 DataSet,DataTable 去逐一對映到指定的類別實在太累太麻煩，開發沒有效率
- 專案太多**花式 SQL Command**
- 太多跨資料庫的資料操作

Repository – 使用 Dapper

底層元件大部分都並非強型別，也不是物件導向設計，資料表也不是，所以在使用上要自行評估開發成本。

以開發「好房網買屋 APP - Web API 專案」的經驗來看**幾乎要重寫原本底層既有的所有程式**。

使用 Dapper 還是有個問題…

- 建立一段 SQL Command，簡單！
- 用 Dapper 去執行這一段 SQL Command，簡單！
- 執行 SQL Command 後要將對映轉換資料的類別？

  如果從資料庫取出的資料欄位少的話… 簡單！

  如果從資料庫取出的資料欄位很~~靠北~~多的話…

  想到要**建立類別**就頭痛！



還能有什麼問題？

- 當系統需要建立**很多類別**而且每個類別的**欄位都很多**的時候，光是建立類別的屬性就頭痛，更別說資料對應轉換
- 當Table的欄位有異動時，對映的類別就必須大費周章修改，如果異動欄位多的話，接下來的處理就要花更多的時間



從一段 SQL Command 的查詢建立類別

一段 SQL Command 就要建立一個類別，

一堆 SQL Command 就要建立一堆類別，

想到這邊就開始猶豫～

如果這一堆的 SQL Command 有很多很多的欄位，

就表示要在新建立的類別裡慢慢刻那一堆的屬性

（我還沒說連資料型別也必須對映）

想到這邊還是……***乖乖回去用 DataSet, DataTable***

工程師的美德

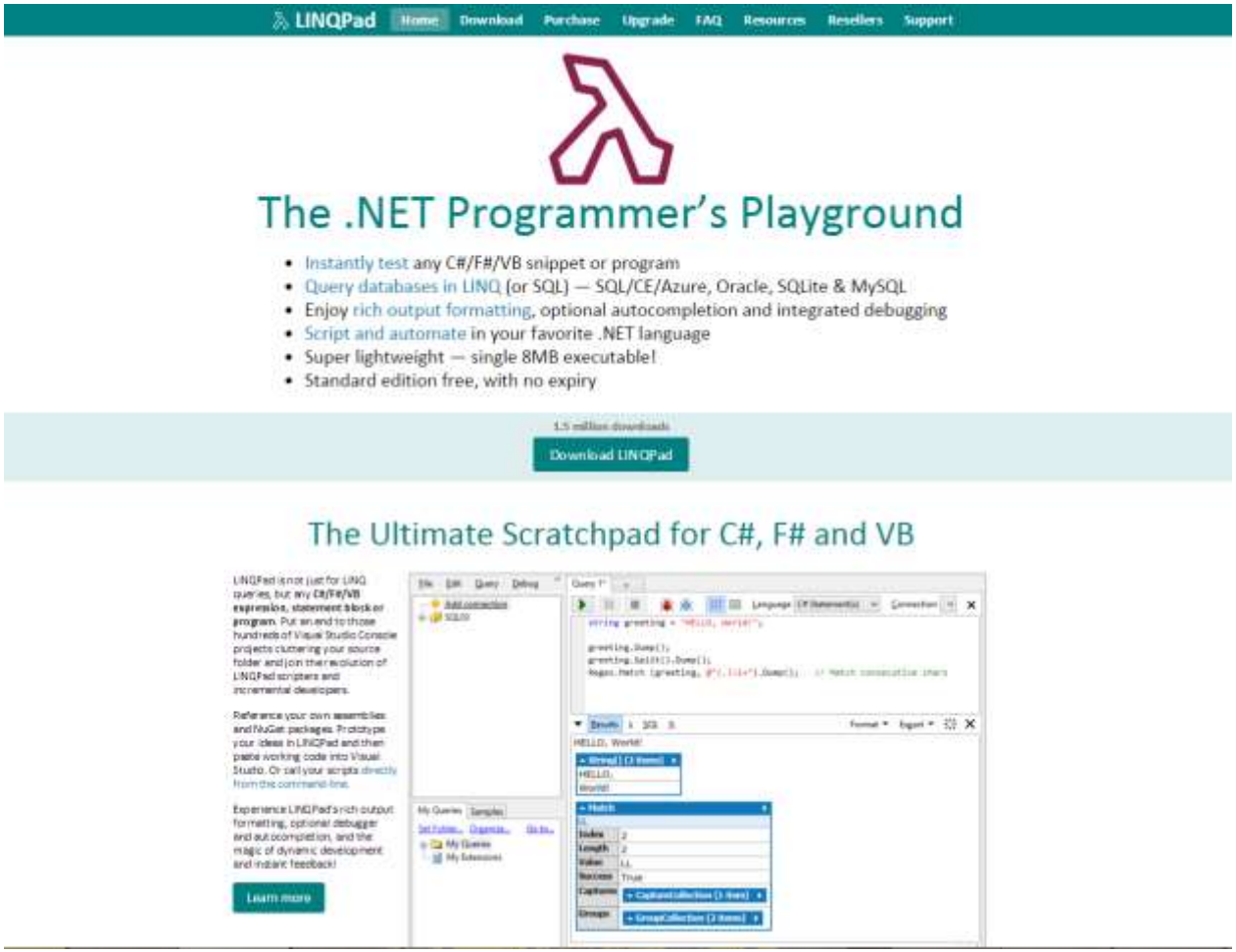


工程師的美德

- 懶，不是遇到麻煩就退縮
- 懶，不是遇到問題就保守
- 懶，不是永遠故步自封
- 懶，要找出一個更好、更快、更簡單的處理做法
- 懶，要找出一個聰明做事的方法，取代繁瑣無趣的操作

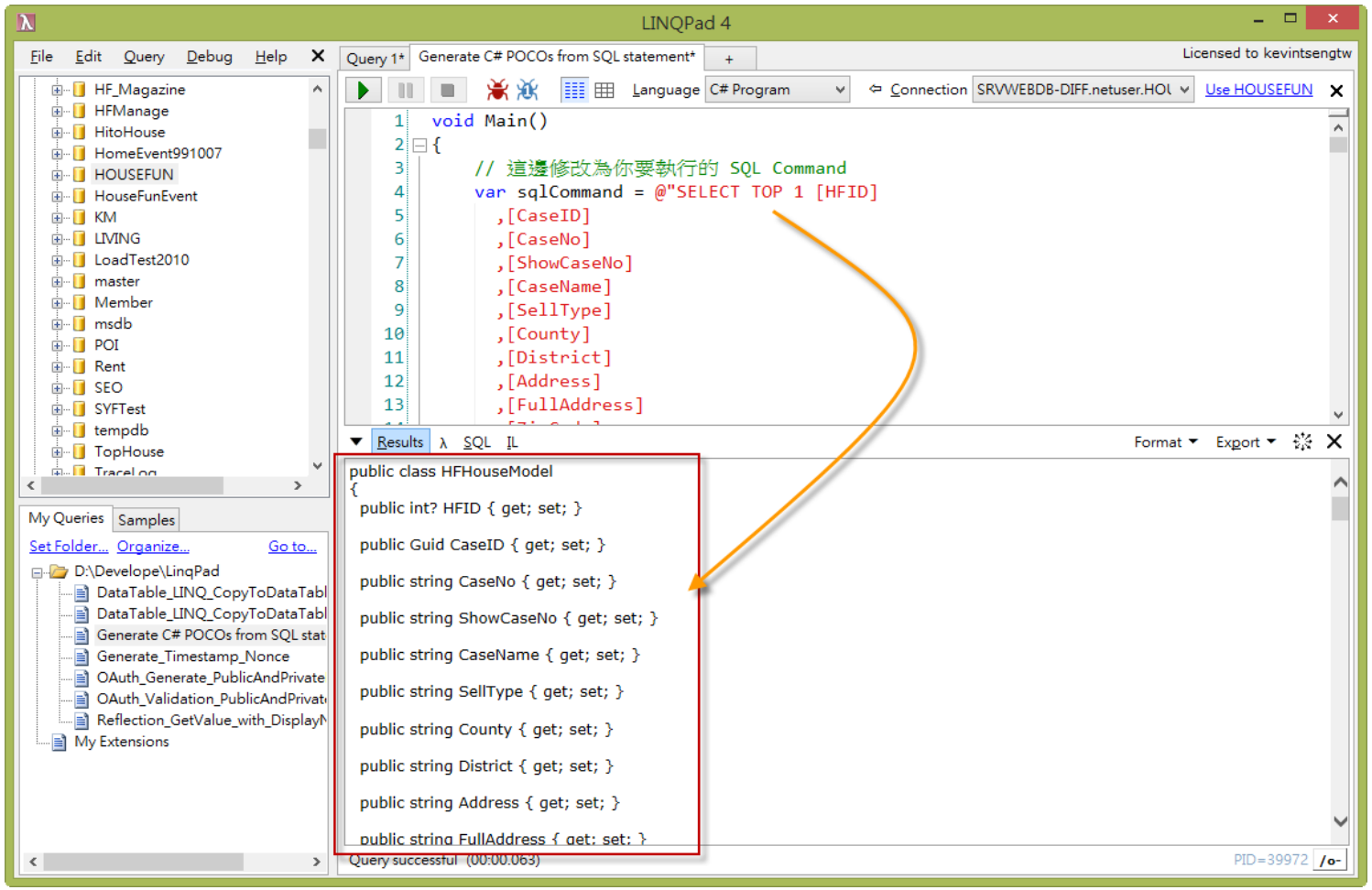


當然有快速又簡單的做法啦 ~



http://www.linqpad.net/

使用 LINQPad 快速產生 SQL Command 對映的類別





demo

使用 LINQPad
快速產生 SQL Command 對映的類別

使用 Dapper

- 依然可以保持使用 T-SQL 操作資料的做法 - **彈性**
- 簡化資料對映轉換的程序 - **簡單**
- 配合 LINQPad 的操作，可快速建立對映的類別 - **快速**

- 如果再加上良好的架構、職責單一且分離的設計，
  可以達成實作單元測試的建立 - **品質與驗證**

使用 Dapper + LINQPad

- 公司裡有些同事改用 Dapper + LINQPad 之後，

  原本需要**三小時**處理的工作，現在只需要花**三分鐘**就解決了

- 多出來的時間就可以 ~~上臉書、訂團購、等下班~~ 學習與進步



使用 Dapper

使用 Dapper 是可以解決原本使用弱型別的型別轉換問題，

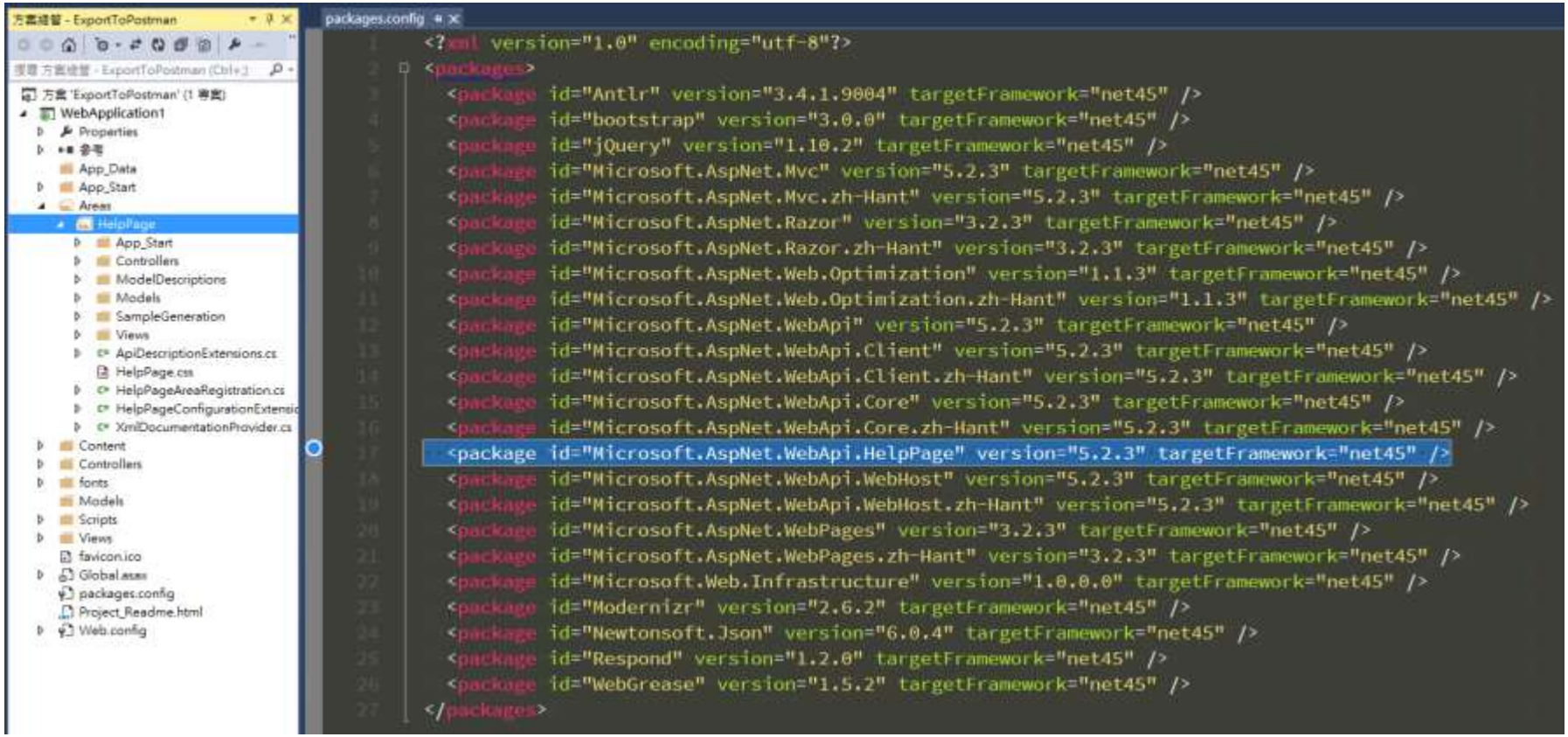
對於專案裡是否要完全使用強型別，

要看專案本身的架構而定，

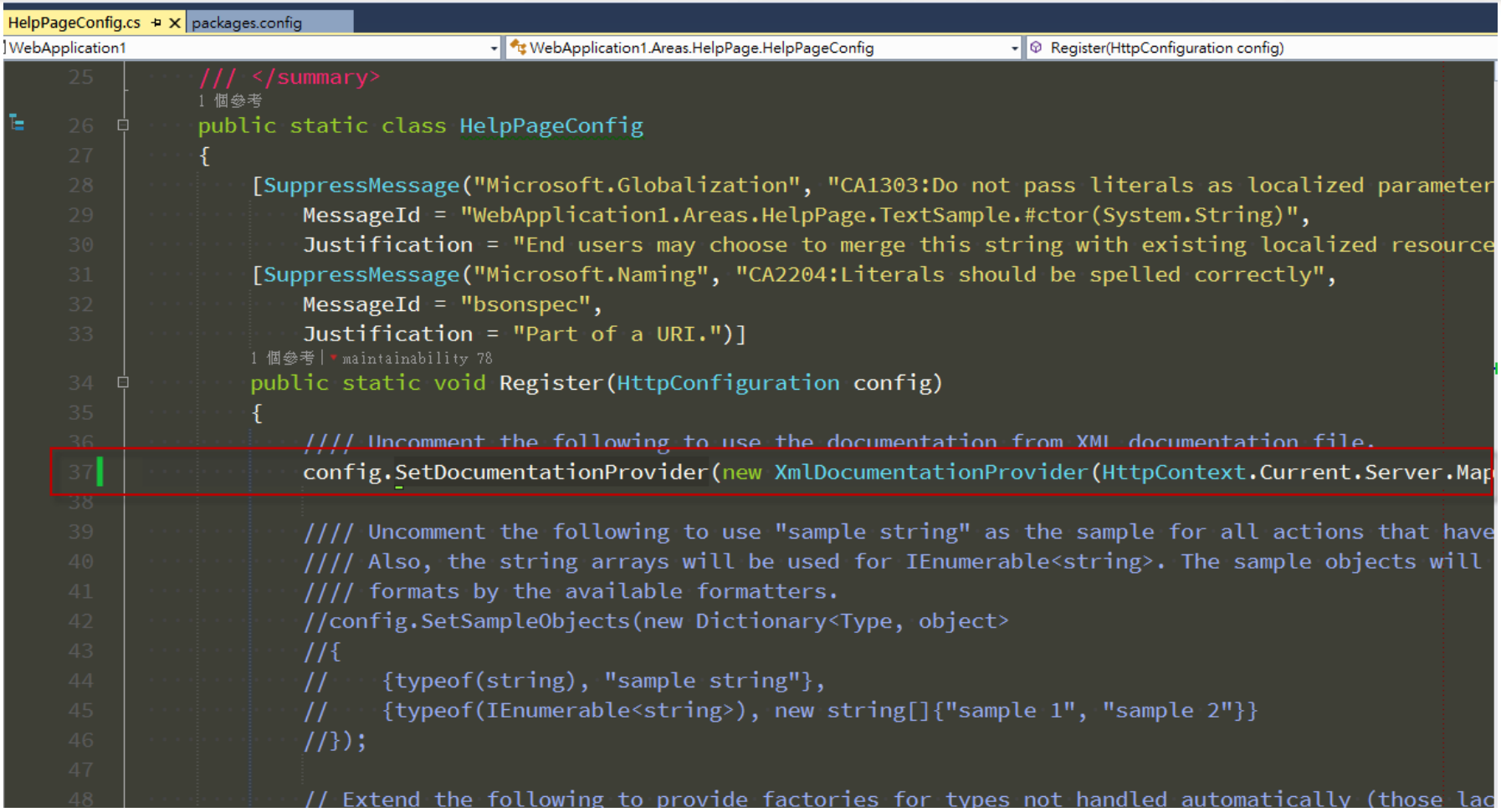
不要因為硬要使用強型別就完全摒除弱型別



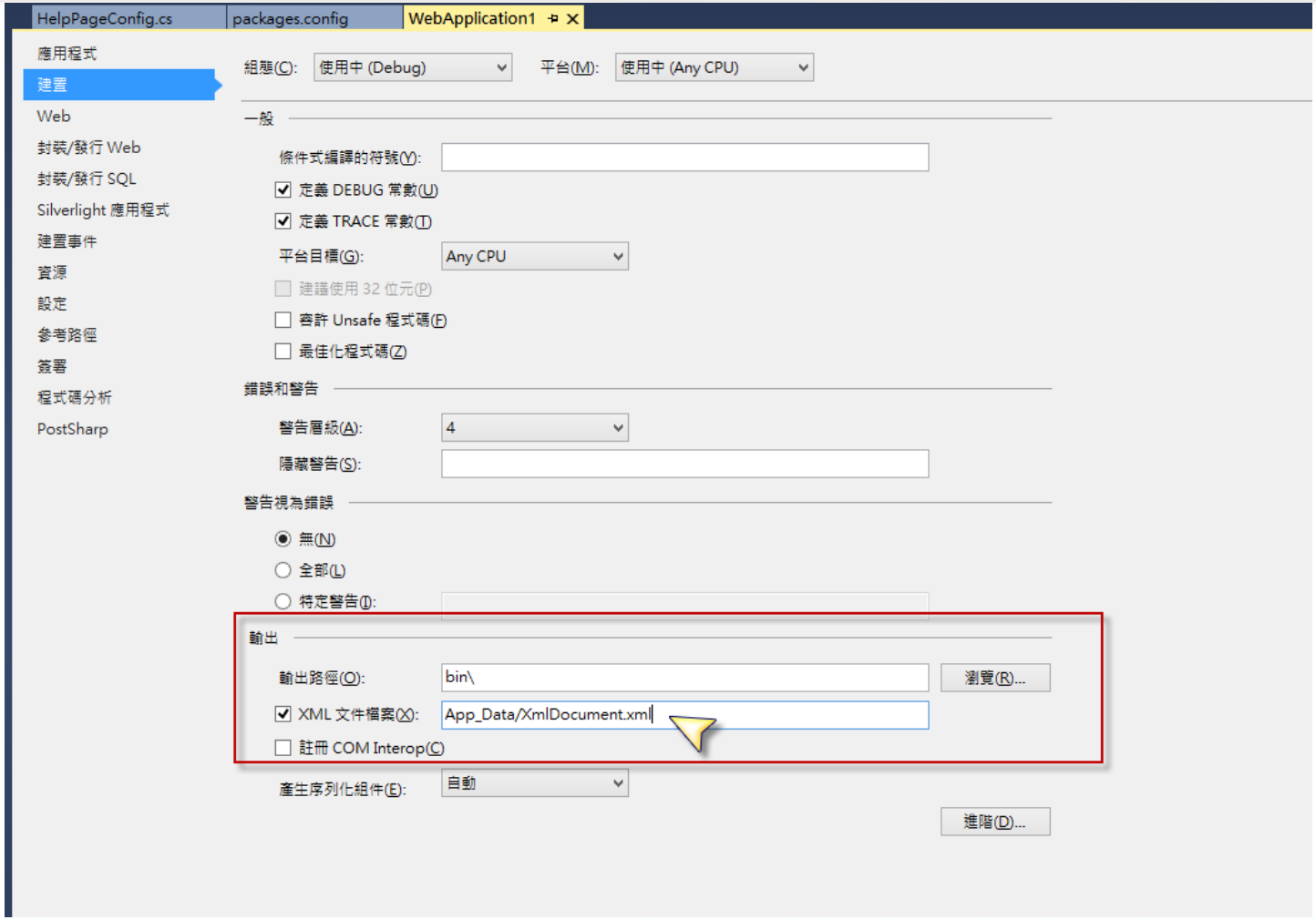
ASP.NET Web Api Help Page



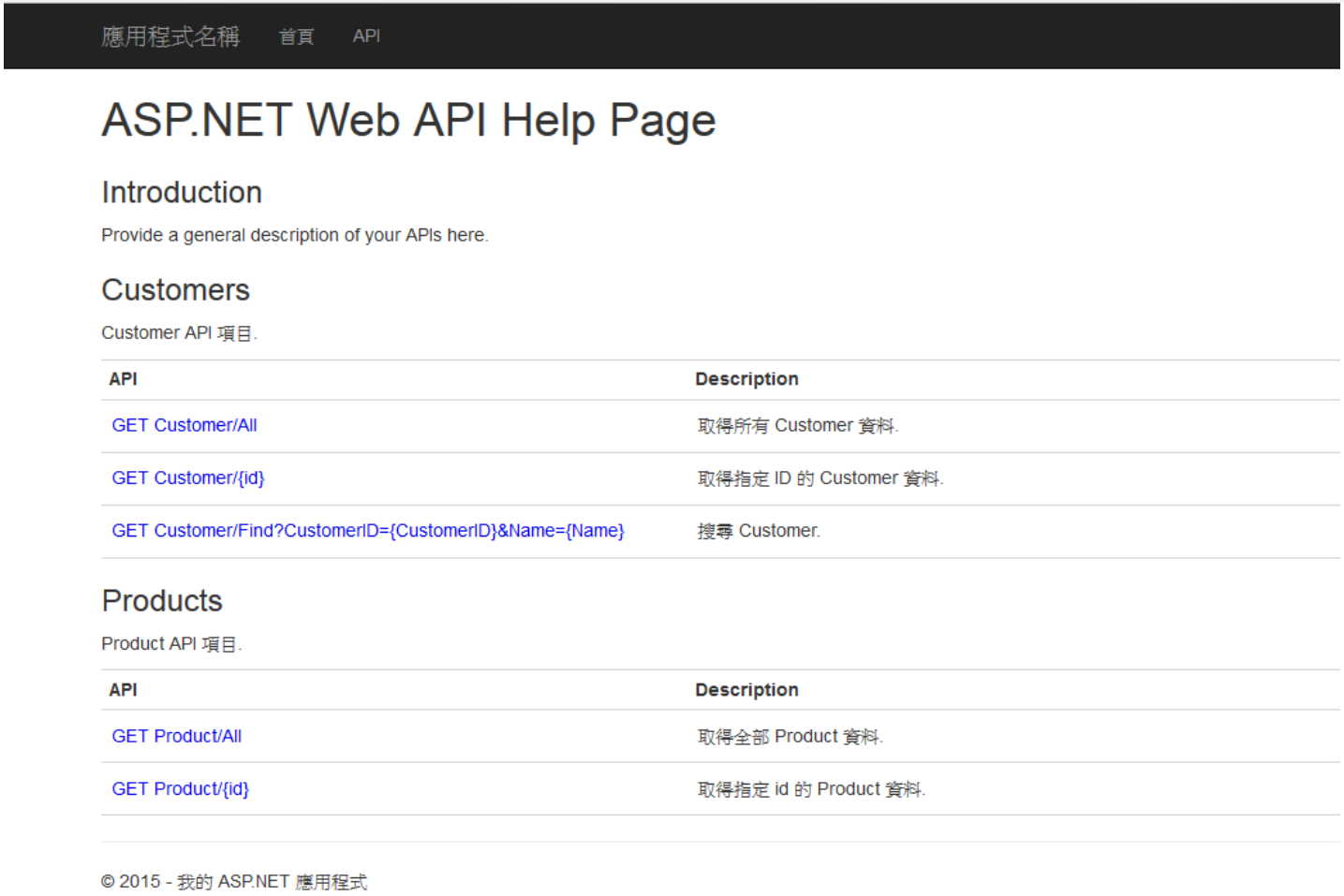
ASP.NET Web Api Help Page



ASP.NET Web Api Help Page



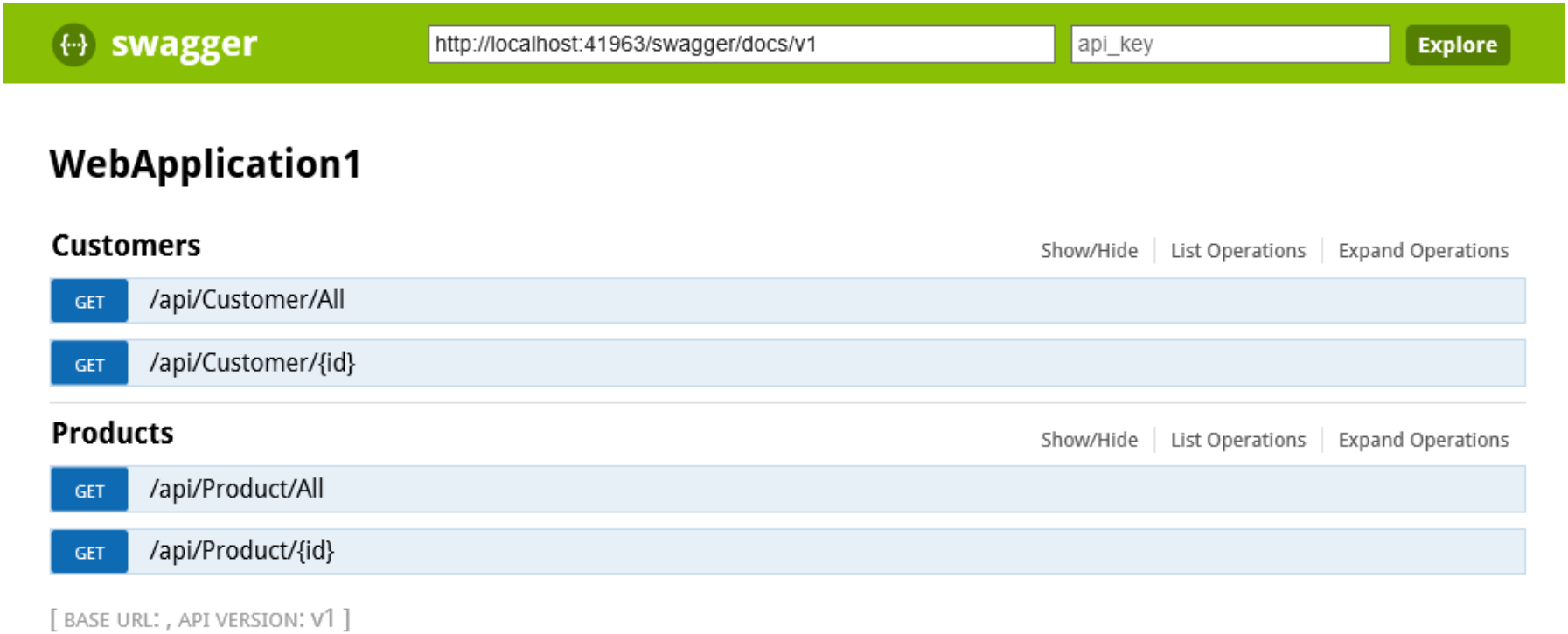
ASP.NET Web Api Help Page



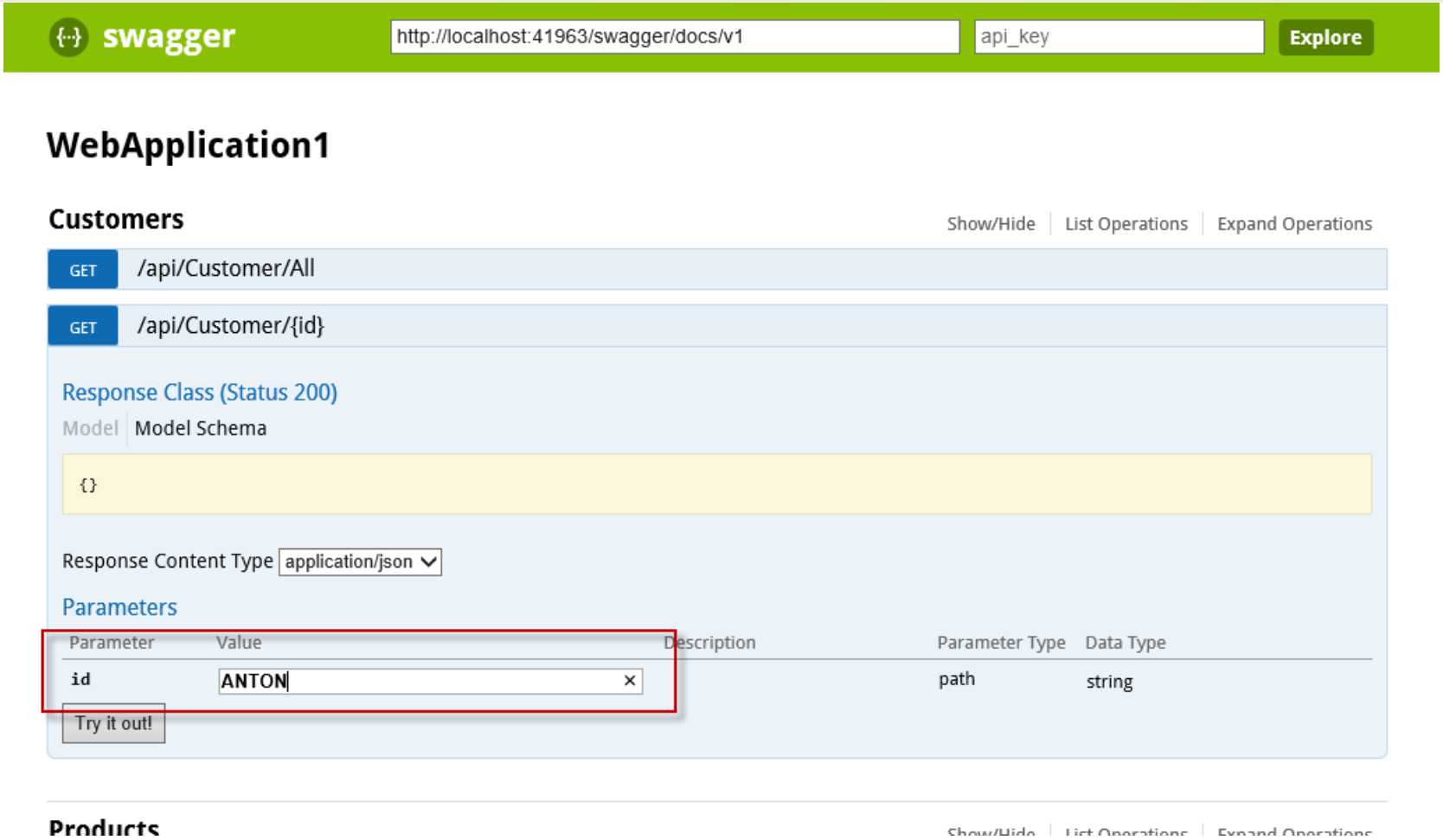
Web API Test Client

- A simple Test Client built on top of ASP.NET Web API Help Page
- yaohuang
  https://github.com/yaohuang/WebApiTestClient
- wuchang
  https://github.com/wuchang/WebApiTestClient

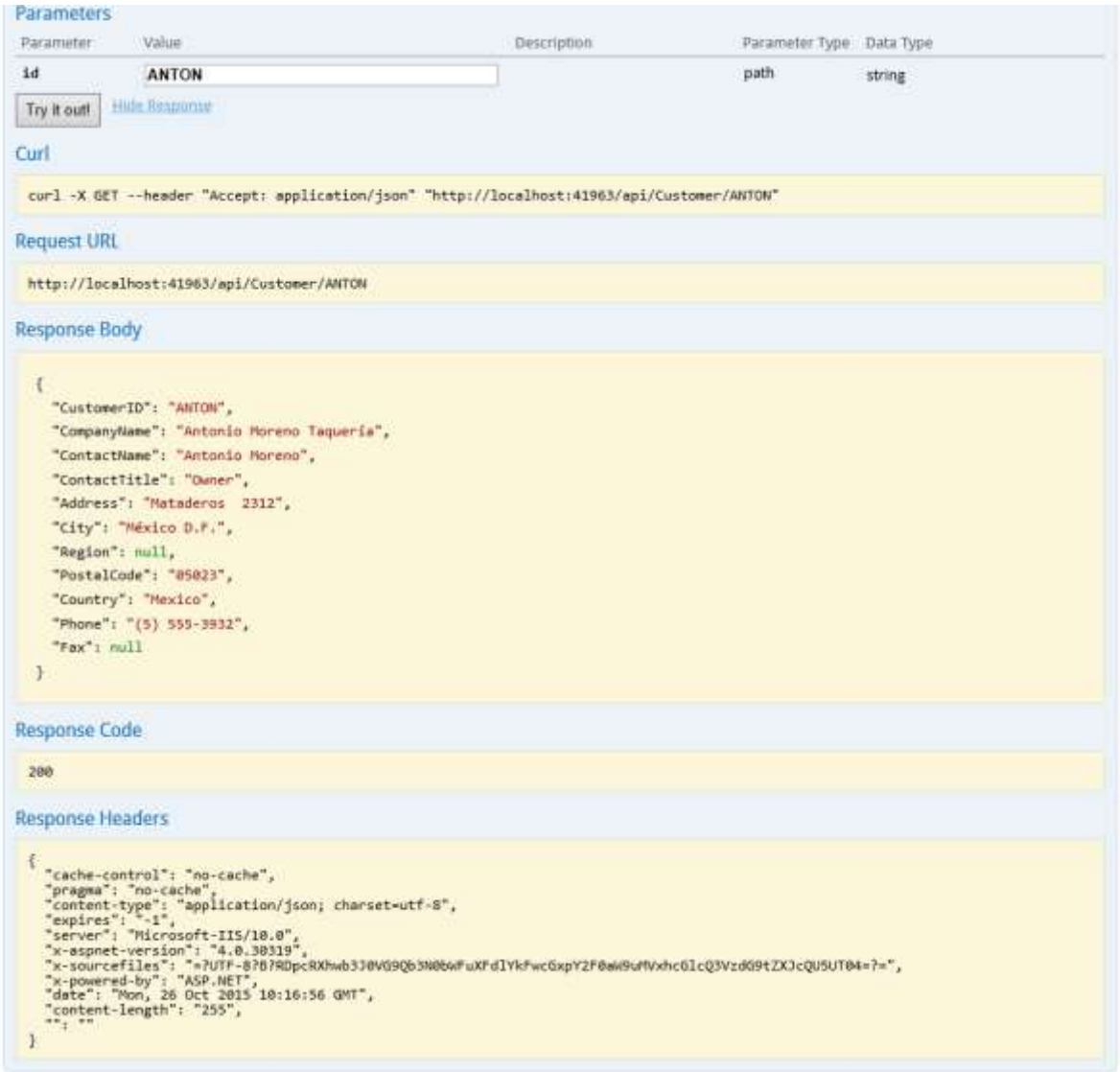
Swagger


swagger
http://localhost:41963/swagger/docs/v1
api_key
Explore
WebApplication1
Customers
Show/Hide
List Operations
Expand Operations
GET
/api/Customer/All
GET
/api/Customer/{id}
Products
Show/Hide
List Operations
Expand Operations
GET
/api/Product/All
GET
/api/Product/{id}
[BASE URL: , API VERSION: V1]

Swagger


swagger
http://localhost:41963/swagger/docs/v1
api_key
Explore
WebApplication1
Customers
Show/Hide
List Operations
Expand Operations
GET
/api/Customer/All
GET
/api/Customer/{id}
Response Class (Status 200)
Model
Model Schema
{}
Response Content Type
application/json
Parameters
Parameter
Value
Description
Parameter Type
Data Type
id
ANTON
path
string
Try it out!
Products

Swagger



Parameters

Parameter	Value	Description	Parameter Type	Data Type
id	ANTON		path	string

Try it out! Hide Response

Curl

```
curl -X GET --header "Accept: application/json" "http://localhost:41963/api/Customer/ANTON"
```

Request URL

```
http://localhost:41963/api/Customer/ANTON
```

Response Body


```
{
  "CustomerID": "ANTON",
  "CompanyName": "Antonio Moreno Taquería",
  "ContactName": "Antonio Moreno",
  "ContactTitle": "Owner",
  "Address": "Mataderos  2312",
  "City": "México D.F.",
  "Region": null,
  "PostalCode": "05023",
  "Country": "Mexico",
  "Phone": "(5) 555-3932",
  "Fax": null
}
```


Response Code

```
200
```

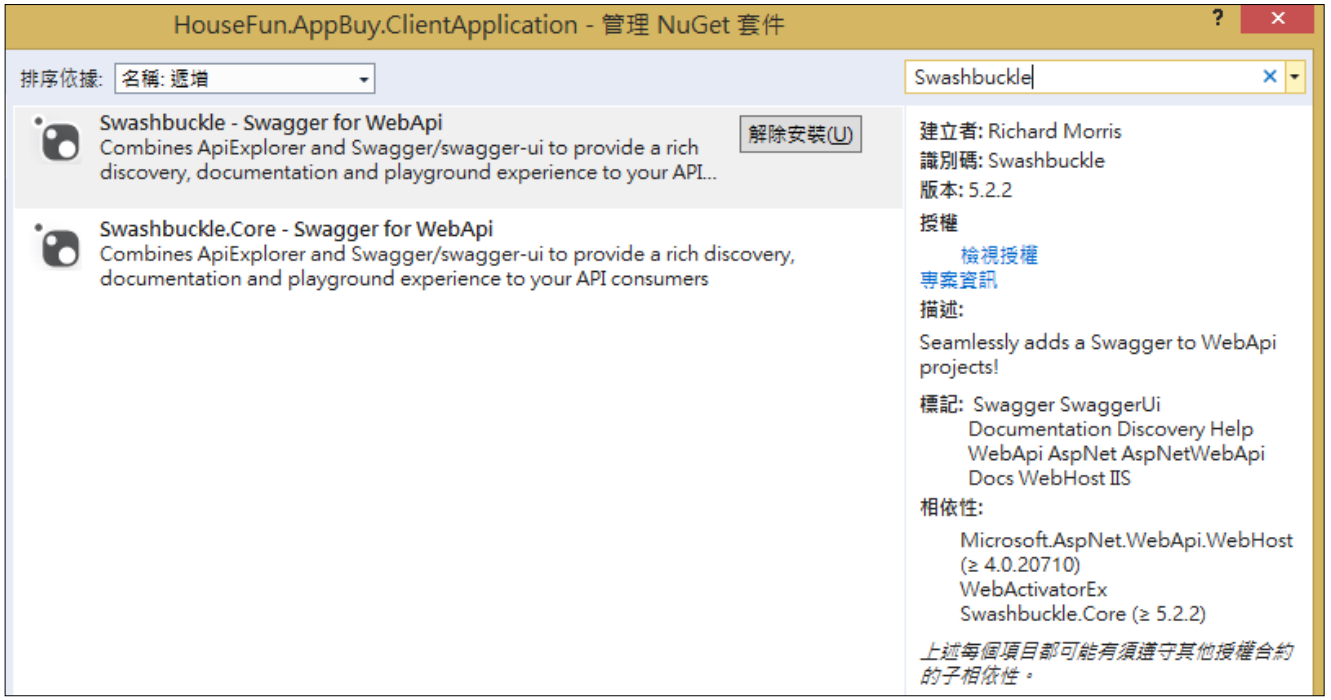
Response Headers


```
{
  "cache-control": "no-cache",
  "pragma": "no-cache",
  "content-type": "application/json; charset=utf-8",
  "expires": "-1",
  "server": "Microsoft-IIS/10.0",
  "x-aspnet-version": "4.0.30319",
  "x-sourcefiles": "=?UTF-8?B?RDpcRXhwb3J0VG9Qb3N0bWFuXFdlYkFwcGxpY2F0aW9uMVxhc0lcQ3VzdG9tZXJcQU5UT04=?=",
  "x-powered-by": "ASP.NET",
  "date": "Mon, 26 Oct 2015 10:16:56 GMT",
  "content-length": "255",
  "": ""
}
```

ASP.NET Web API + Swagger

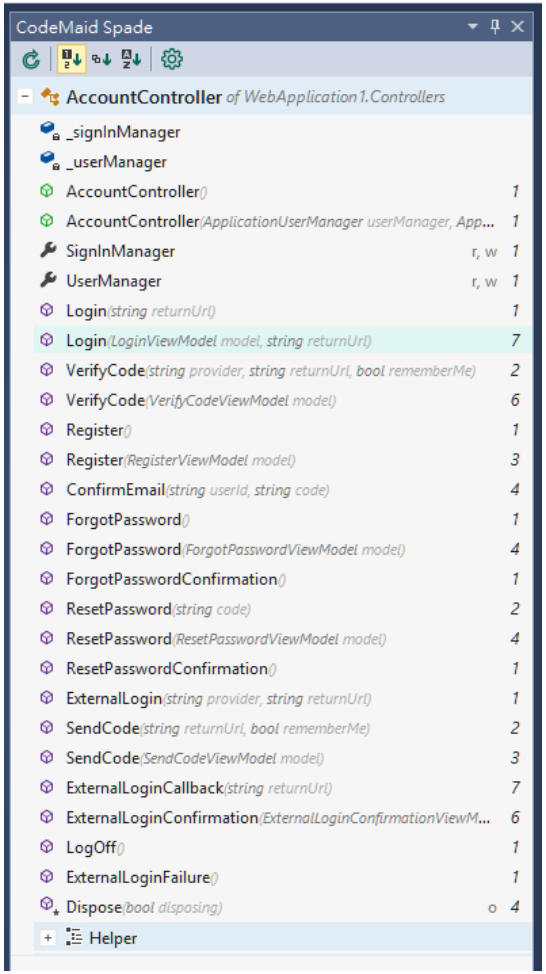
- Swashbuckle – Swagger for WebApi

  Swashbuckle.Core – Swagger for WebApi

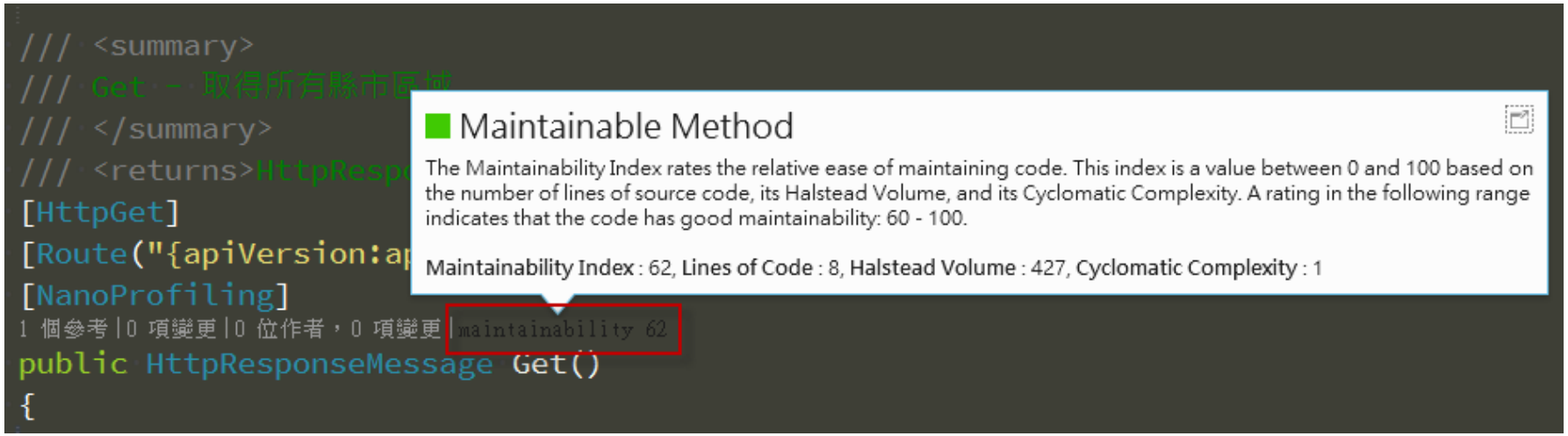


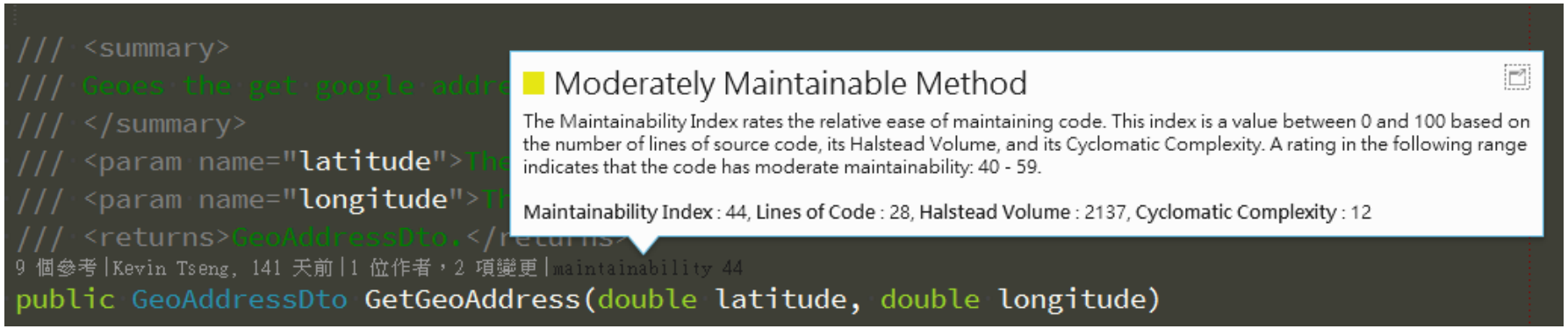
觀察程式碼複雜度 – 使用 CodeMaid



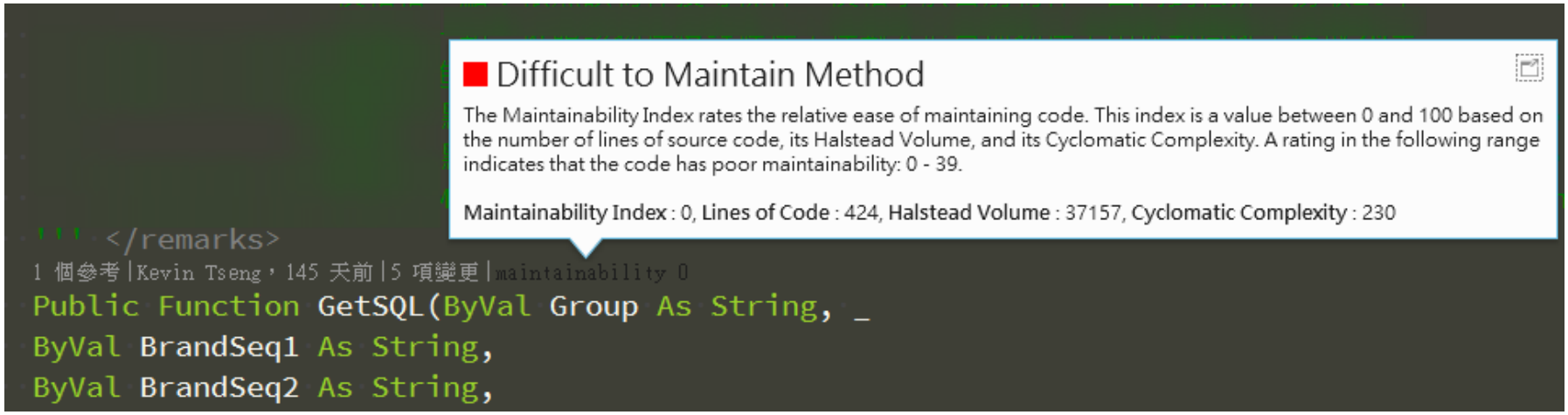


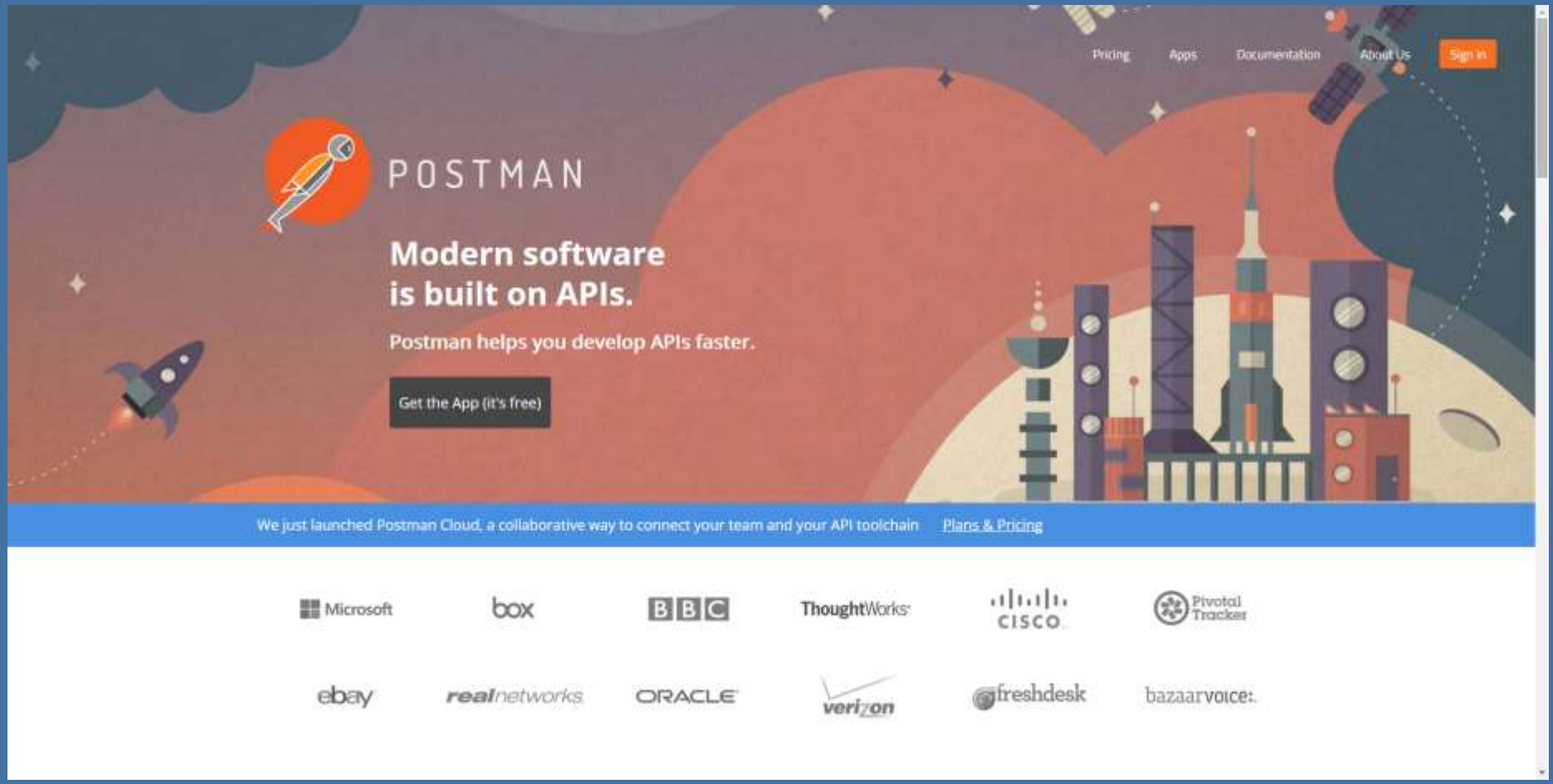
Microsoft CodeLens Code Health Indicator





Microsoft CodeLens Code Health Indicator




Pricing
Apps
Documentation
About Us
Sign in
POSTMAN
Modern software
is built on APIs.
Postman helps you develop APIs faster.
Get the App (it's free)
We just launched Postman Cloud, a collaborative way to connect your team and your API toolchain
Plans & Pricing
Microsoft
box
BBC
ThoughtWorks
CISCO
Pivotal Tracker
ebay
realnetworks
ORACLE
verizon
freshdesk
bazaarvoice

Postman

Client 端的工具建議使用 Postman App

https://www.getpostman.com/

http://www.getpostman.com/docs/

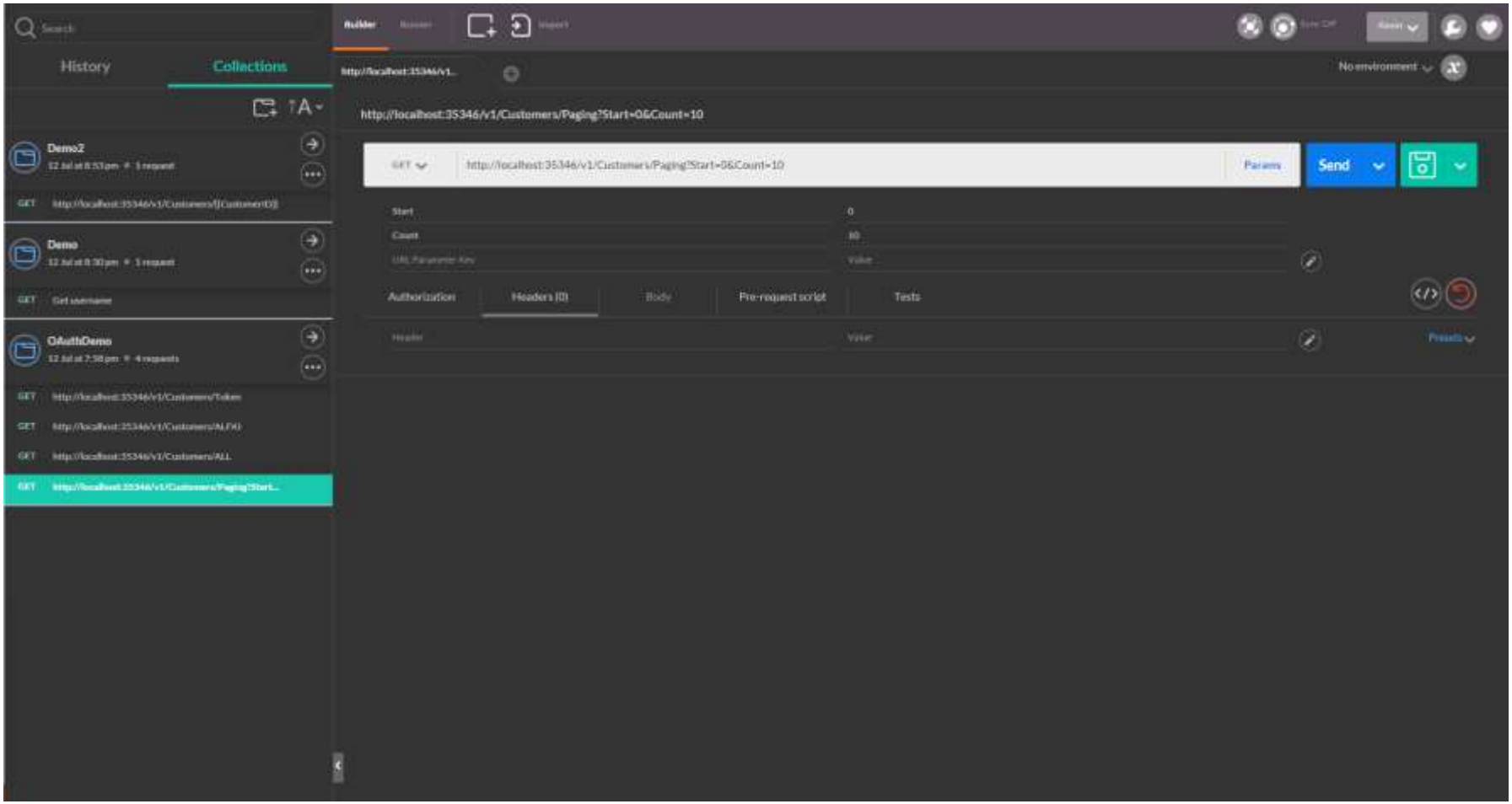
用 Postman 的好處：

可快速建立 Api 的測試情境，方便作管理

新版的功能有增加多個頁籤

可以依據情境、環境建立不同的參數和數據做切換使用

Postman


History
Collections
Builder
Demo2
Demo
OAuthDemo
http://localhost:35346/v1/Customers/Paging?Start=0&Count=10
Send
Params
Start
Count
Authorization
Headers (0)
Body
Pre-request script
Tests



demo

Postman 操作示範

Postman

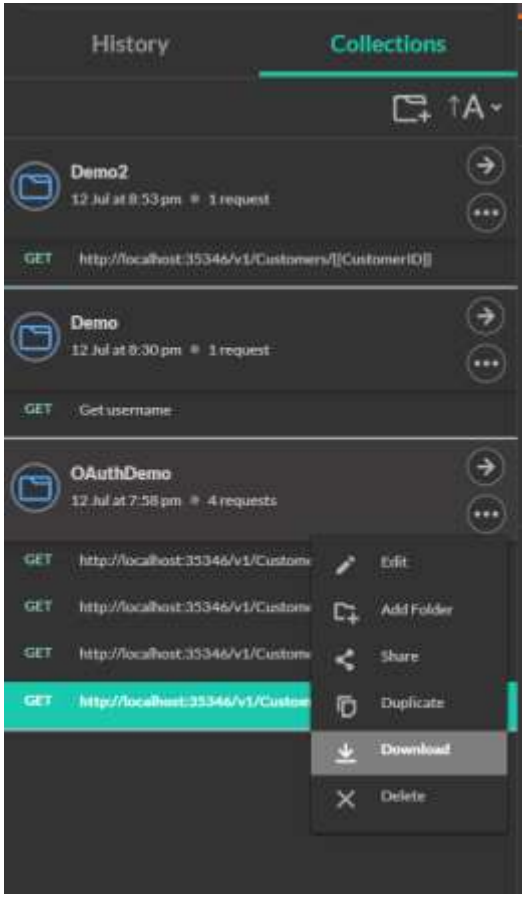
- 如果開發的 API 項目很多，在 Postman 的 API 項目列表就會有一大堆
- 一個人開發時，還不會有問題。

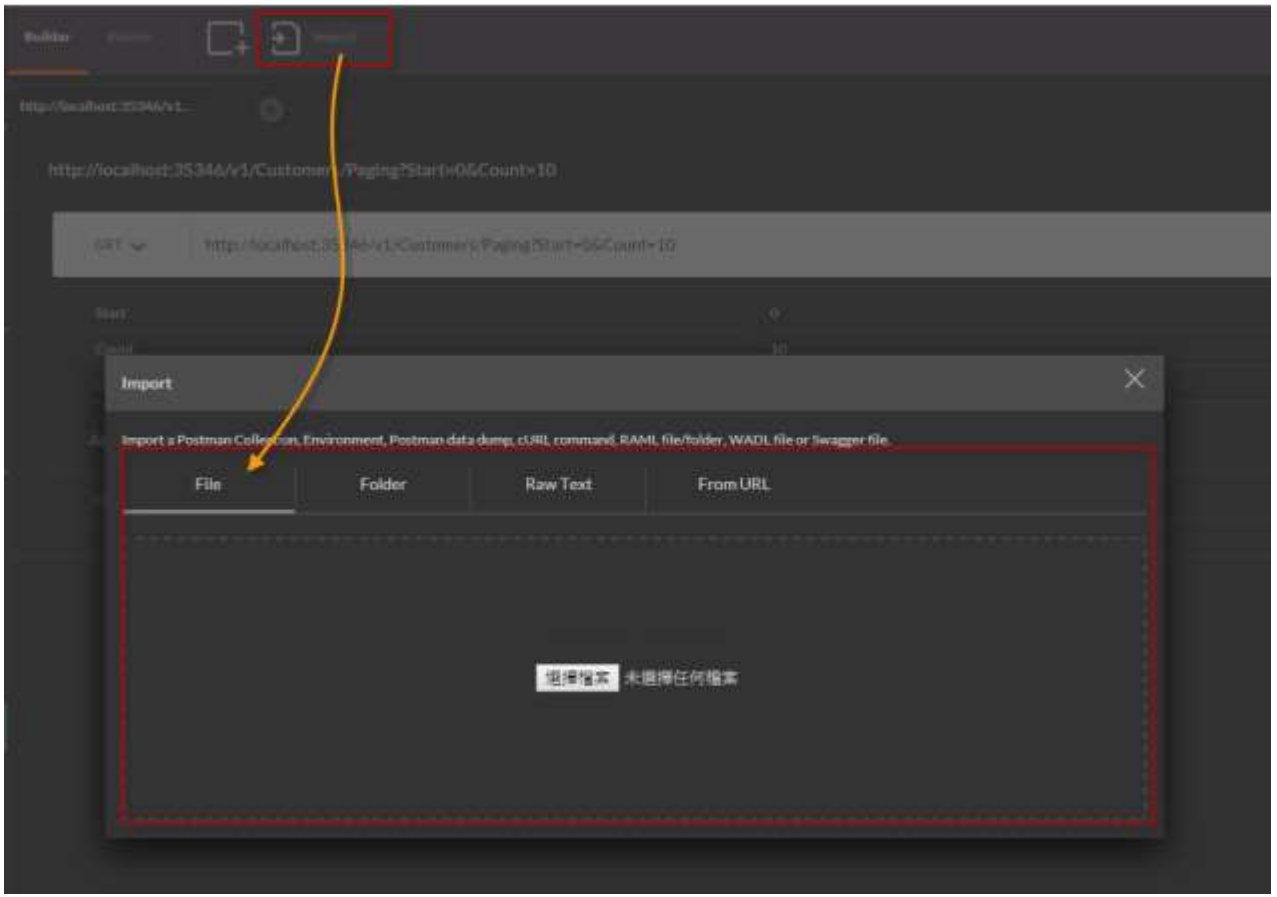
  多人共同開發就必須要保持 API List 內容的一致，

  所以可以由 Team Leader 做 API List 的管理，

  然後使用匯出的功能，提供給團隊成員匯入

Postman


History
Collections
Demo2
12 Jul at 8:53 pm • 1 request
GET http://localhost:35346/v1/Customers/{{CustomerID}}
Demo
12 Jul at 8:30 pm • 1 request
GET Get username
OAuthDemo
12 Jul at 7:58 pm • 4 requests
GET http://localhost:35346/v1/Custom
GET http://localhost:35346/v1/Custom
GET http://localhost:35346/v1/Custom
GET http://localhost:35346/v1/Custom
Edit
Add Folder
Share
Duplicate
Download
Delete


Import
Import a Postman Collection, Environment, Postman data dump, cURL command, RAML file/folder, WADL file or Swagger file.
File
Folder
Raw Text
From URL
選擇檔案
未選擇任何檔案

Postman 裡面的 API 內容不一致

當 Web Api 專案所建立的 Controller 類別與 Action 方法很多的時候，在 Postman 就必須一個一個去建立對映的執行情境，操作會很繁瑣

當多人開發時，版本更動會相當頻繁，

團隊開發成員所使用的執行情境就會無法一致

Postman

如果版本更新相當頻繁而且東西又多的話，那 Team Leader 成天只要維護 API List 的資料，其他事情就可以不用做了

所以必須要找出一個更為簡單且快速的替代方案，
讓團隊可以在更新專案原始碼後就可以取得最新的 API List，並且匯入 Postman

其實方法很早已前就有啦，
透過 ASP.NET Web API Help Page 的功能來完成

Web Api List 匯入 Postman

- 在 Web Api 專案內建立一支 API，

  將專案內的所有公開的 API 匯出成 Postman 格式，

  可以解決在 Postaman 手動新增執行情境的操作，

  以及解決各成員操作的 API 內容不一致的問題



demo

Web Api List 匯入 Postman

Web Api List 匯入 Postman

- 可以提供給協同開發的其他開發團隊，

  讓所有開發團隊都使用相同的 API List

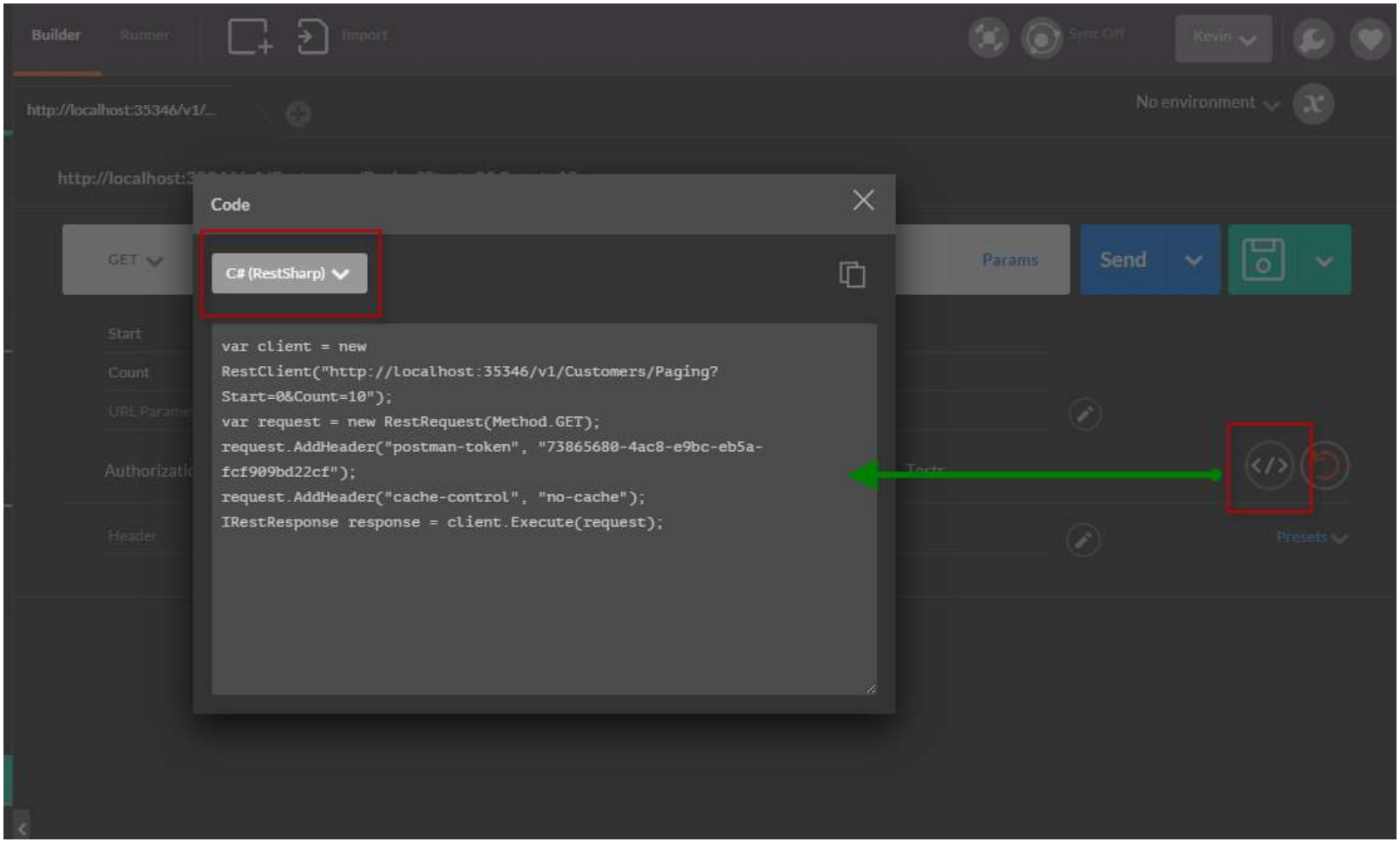
- **問過我們家的 App 開發工程師，他們真的很需要這個功能**



Postman – Code Snippets

- Postman 不只是單純的去測試我們開發的 API，
  Postman 還可以產出 C# 的程式，
  我們可以將這段程式複製起來然後貼到開發的專案裡執行
  （有些專案是需要在程式裡去呼叫使用其他的 Web API）

Postman – Code Snippets





demo

LINQPad 執行
Postman Code Snippets 程式

Postman – Code Snippets

- Postman 所產出的 C# 程式碼可以馬上貼到 LINQPad 做執行，不需要再去開啟 Visual Studio 或是貼到正在開發的專案裡做測試
- 從 Postman 裡馬上產出程式碼，立刻複製然後貼到 LINQPad 裡做執行，馬上看結果

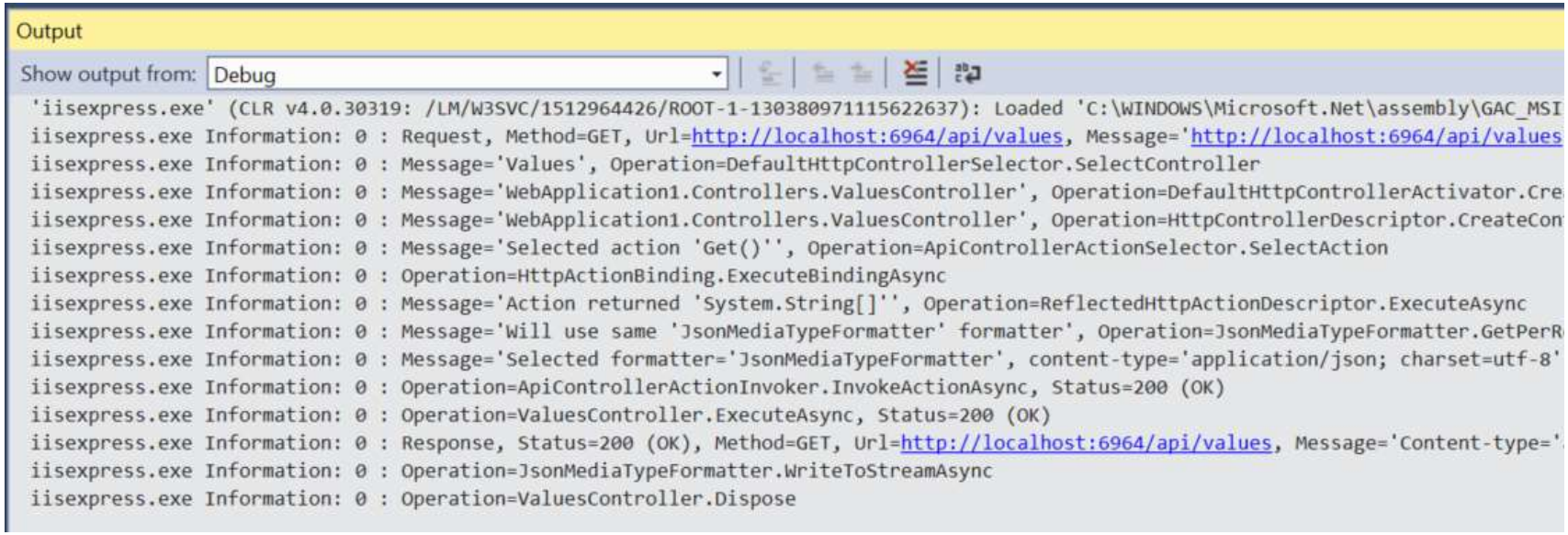
監控與記錄

監控與記錄的第三方套件

- Microsoft.AspNet.WebApi.Tracing
- Elmah
- NLog
- NanoProfiler

Microsoft.AspNet.WebApi.Tracing

- 開發時的好幫手



http://www.asp.net/web-api/overview/testing-and-debugging/tracing-in-aspnet-web-api

Elmah

- 這是一定要裝的，記錄系統的異常例外狀況

  很久以前，一直到現在都強調 Elmah 是一定要裝

- twMVC#2 ASP.NET MVC 偵測監控與 Log 記錄



NLog

- 用來記錄 Request 的輸入、輸出的所有資料
- 用來記錄 Request 在發生錯誤時的所有資料
- 用來記錄程式裡想要追蹤觀察的訊息



http://nlog-project.org/

NLog + MessageHandler

- [筆記][WebAPI] Message Handler 初體驗-搭配 NLog 記錄所有 WebAPI 的 Request 與 Response | topcat 姍舞之間的極度凝聚 - 點部落

Profiling?

- 如何偵測監控一個 Request 進來到輸出的這段時間所經過各個程式的執行時間？
- 用 StopWatch？別鬧了啦
- 用 NLog 去記錄？你是認真的嗎？
- ASP.NET MVC 可以使用 MiniProfiler 或 Glimpse，但是 ASP.NET Web API 並沒有頁面可以顯示……

Profiling?

- twMVC#15 – 開發的效能與效率

  這一場有介紹過 MiniProfiler 與 Glimpse

  (如果你還有印象的話)



https://docs.com/is-twMVC/2591/twmvc-15

NanoProfiler

- NanoProfiler 是一個 EF Learning Labs 出品的免費性能監控類庫
- 使用方式與做法與 MiniProfiler 類似，但設計理念有較大差異
- 可提供即時性能監控數據

  可對資料庫存取進行監控

  提供清楚、簡單的數據顯示頁面，方便查看

目前還需要改善的地方

- Elmah 與 NLog 因為 DBA 不希望這些 Log 資料儲存到資料庫裡，所以目前都還是儲存在一個個的文字檔裡
- 檔案有儲存問題（曾經有一個月的 Log 檔案沒有清掉，而讓 Server 硬碟空間拉警報）
- NLB 機制下的各台 Server 都有各自的 Log 檔案（管理與搜尋不易）
- 無法使用第三方服務替代（Elmah.io or Loggly…）
- Log 可改為使用 Exceptionless 機制（可搭配 NLog）

目前還需要改善的地方

- NanoProfiler 也一樣面臨 Log 儲存的問題

  所以目前仍然是使用預設 Log 資料存在記憶體

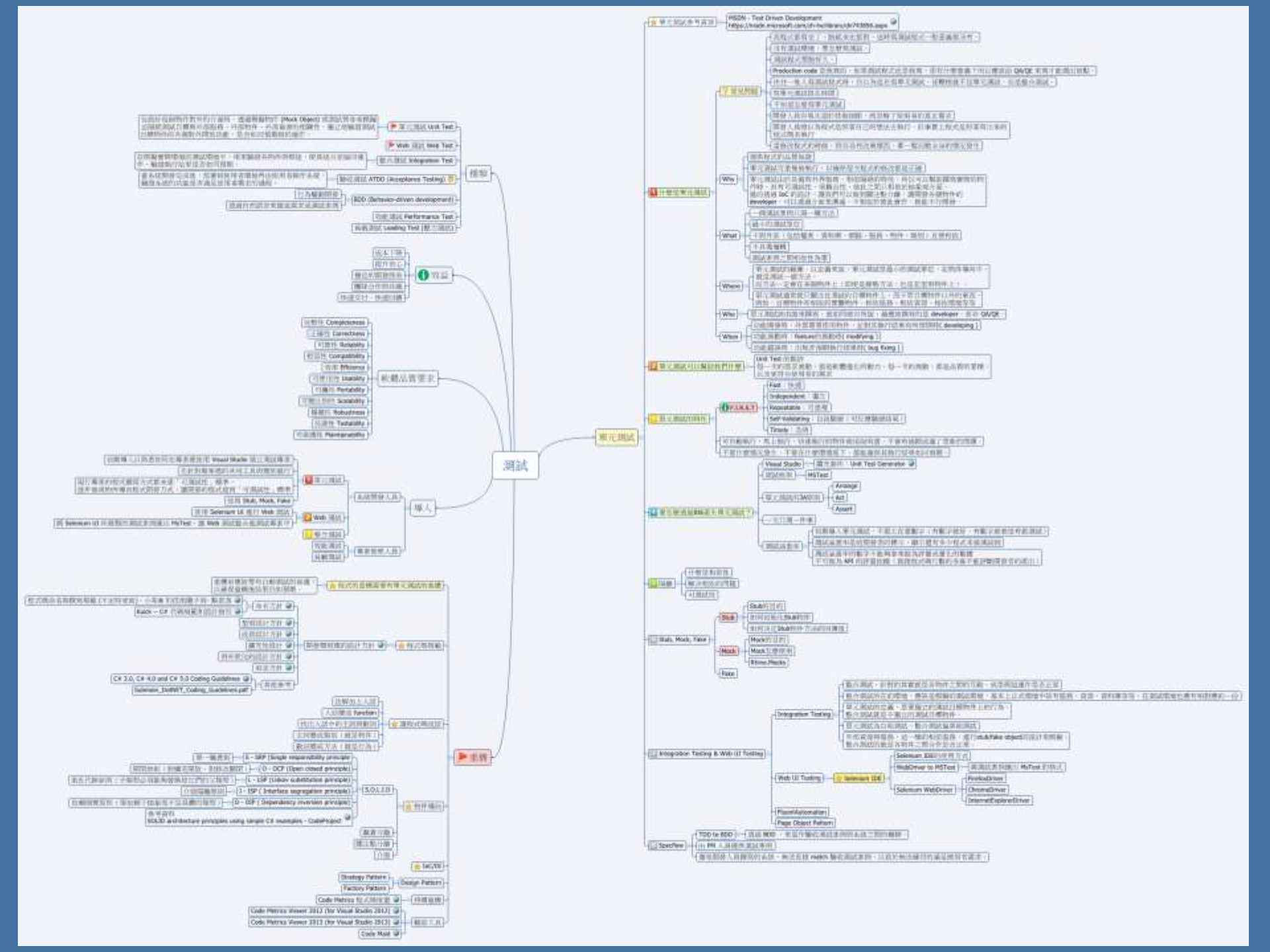
- Log 存在記憶體的數量還是有限制的，

  現在是只有存放最新 200 筆的記錄，資料無法保留下來

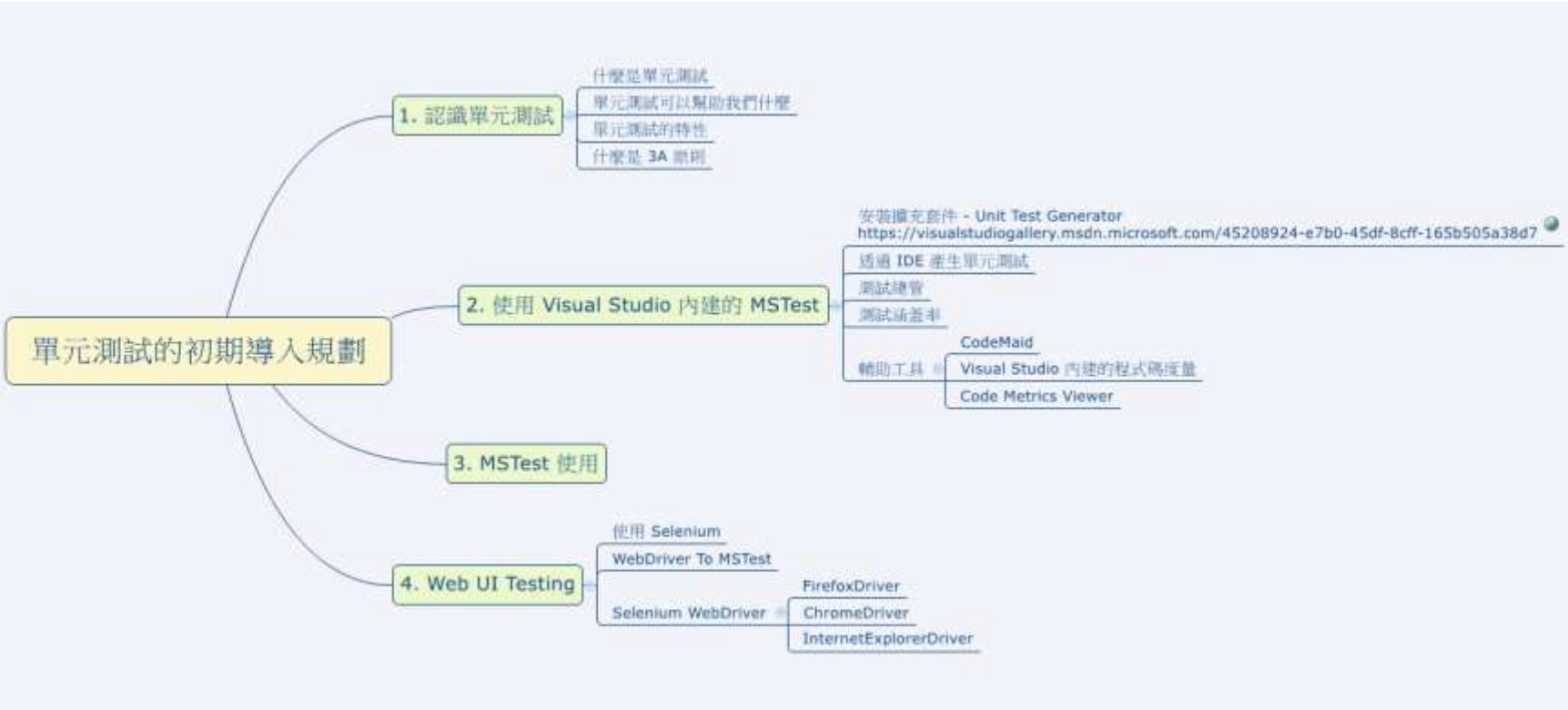
- NanoProfiler 有提供支援 ELK

  ElasticSearch + Logstash + Kibana

單元測試與整合測試


測試
概型
單元測試 Unit Test
Web 測試 Web Test
整合測試 Integration Test
驗收測試 ATDD (Acceptance Testing)
BDD (Behavior-driven development)
功能測試 Performance Test
負載測試 Loading Test (壓力測試)
可讀
可維護
可擴展
軟體品質要求
Completeness
Correctness
Reliability
Compatibility
Efficiency
Usability
Portability
Scalability
Robustness
Testability
Maintainability
導入
單元測試
Web 測試
整合測試
系統開發人員
專案管理人員
Visual Studio
Stub, Mock, Fake
Selenium IDE
Selenium WebDriver
MsTest
重構
程式的基礎需要有單元測試的基礎
程式碼設計
Code Metrics
Design Pattern
視覺工具
Code Metrix Viewer 2013 (for Visual Studio 2012)
Code Metrix Viewer 2013 (for Visual Studio 2013)
Code Maid
單元測試
參考資源
MSDN - Test Driven Development
計劃單元測試
常見問題
Why
What
Where
Who
When
單元測試可以幫助我們什麼
Unit Test 的設計
單元測試的特性
F.I.R.S.T.
Fast
Independent
Repeatable
Self-Validating
Timely
單元測試的寫法
Visual Studio
測試框架
MSTest
單元測試的3A原則
Arrange
Act
Assert
一次只測一件事
測試命名
Stub, Mock, Fake
Stub
Mock
Fake
Integration Testing & Web UI Testing
Integration Testing
Web UI Testing
Selenium IDE
Selenium WebDriver
PageAutomation
Page Object Pattern
SpecFlow
TDD to BDD
Gherkin
Step Definitions
Unit Test
Integration Test
Selenium IDE
WebDriver to MSTest
Selenium WebDriver
FirefoxDriver
ChromeDriver
InternetExplorerDriver

內部訓練-單元測試第一階段導入規劃



內部訓練-測試分享課程及其他相關分享

- **Lesson 1：**

  基本單元測試介紹與認識

- **Lesson 2：**

  Unit Test Generator, 認識測試總管, Assertion, Fluent Assertions

- **Lesson 3：**

  物件比較, Private 與 Internal 方法的測試

內部訓練-測試分享課程及其他相關分享

- Lesson 4：

  MsTest Unit Test Event Hook, 日期時間的單元測試

- Lesson 5：

  Isolated Unit Test, 相依介面區分職責,

  使用 NSubstitute

- Lesson 6：

  使用 NSubstitute – Mock object,

  使用 ExpectedObjetcs

內部訓練-測試分享課程及其他相關分享

- Lesson 7：

  TDD (Test-Driven Development)

- Lesson 8：

  重構與測試 – 以 ASP.NET WebForm WebSite Project 專案為例

內部訓練-測試分享課程及其他相關分享

- ASP.NET MVC / Web API 另一種資料存取方式
  – 使用 Dapper 和 LINQPad

- 因為這一次的分享，公司採購了 LINQPad 5 Premiun Enterprise 授權
  (這是一個好工具，在開發、測試時都一定會用到)

內部訓練-測試分享課程及其他相關分享

- ASP.NET Web API 的單元測試和整合測試

  (以好房網買屋 APP Web API 專案為例)

  Import to Postman, Help Page, Swagger

- 使用 SpecFlow, SpecRun, Pickles 產生測試報告

好房網買屋 APP Web API 專案

- 好房網買屋 APP Web API 專案 – Client Application

  Web Api 專案的測試會分成兩種：

  一、對各個程式類別的單元測試

  二、專案啟始後使用工具直接對專案進行測試（人工）

好房網買屋 APP Web API 專案

- Common, Service, ClientApplication (WebApi)

  各個專案裡面的主要流程一定會建立單元測試

  簽入 Git 前都必須要確認所有的單元測試都是通過的

Repository 測試

- Repository 專案（資料存取）的測試也是要做，
  但優先順序是在上述專案之後
- 資料存取層的測試會直接去相依到外部資源，
  因為開發用的資料庫並不是只有一個開發團隊在使用，
  還有其他團隊、其他部門也共同在使用，
  所以「資料存取層的測試專案」必須要使用一個完全乾淨
  且不受影響的資料庫，這邊所使用的是「LocalDB」

Repository 測試

- 「資料存取層的測試專案」所使用的 LocalDB 資料庫
  不需要事先建立（因為會有存取權限的限制），
  從建立 DB Instance、建立 Table、加入測試資料、清除資料、移除 Table、移除 DB Instance 等等等…
  這些都是透過程式去執行 T-SQL 指令碼

Repository 測試

- 約 60 % 的對資料庫操作的程式都可以完成單元測試，

  但還是有 40 % 是無法完成單元測試，

  並非不能，而是在於是否必要(時間成本與開發成本的權衡)

Repository 測試

資料存取層的單元測試其中一個困難點，跨資料庫的操作

以同一條資料庫連線對多個資料庫做資料的查詢

必需要瞭解用 LocalDB 裡去建立多個資料庫不是難事

難的是資料結構與關連和其他相關的所有都必須一模一樣

Repository 測試

目前資料庫是經過長時間下來、無數多個故事的演化

最後我們所看到的，並不是單純的去複製一個測試資料庫就能簡單的解決。

對策：

對於比較困難的資料存取處理，

這部分的單元測試就暫時選擇不去做

專案測試-說明

- 並沒有使用 TDD 而 BDD 只用在整合測試
- 專案在三月底就已經完成 80% 的程式開發
- 一開始的測試放在共用工具程式與 WebApi 專案的主流程
- 七月前都還在做需求變更的程式開發，測試只有少部分進行
- 專案大部分測試（Service, Repository）是在7~9月進行(開發新版本功能與整合會員功能的時候)

專案測試里程碑

- Code Coverage 測試程式碼涵蓋率

2015-05-20：18.14% (245 個單元測試)

2015-06-16：30.56%

2015-06-24：33.37% (正式版上線前)

2015-09-25：70.15% (v1.1 會員功能)

2015-10-21：69.98%

2015-11-17：77.41%

2015-11-20：77.45%

2015-12-07：77.46% (1310 個單元測試)

不同階段的測試歷程

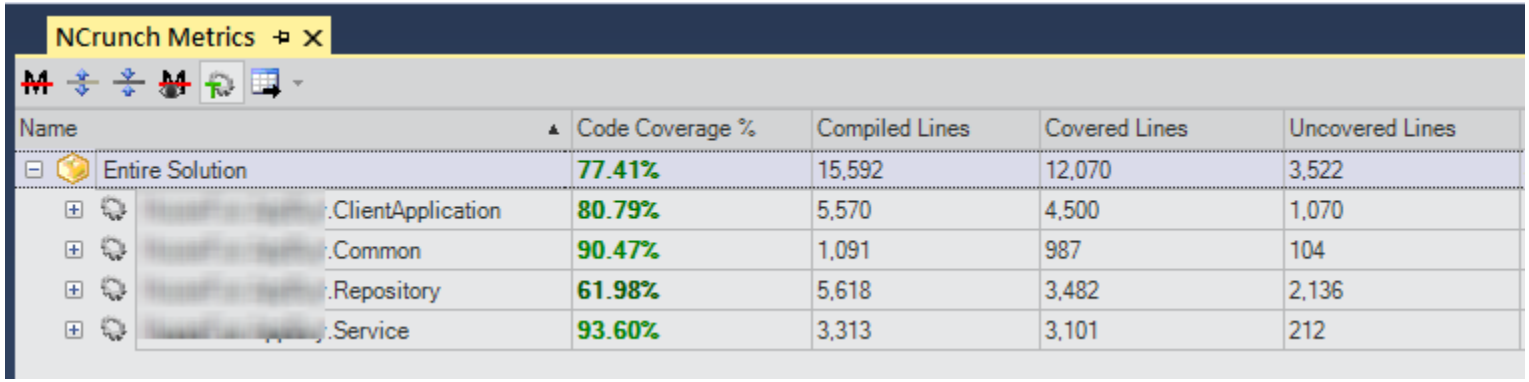
- 2015-10-21

NCrunch Metrics

Name	Code Coverage %	Compiled Lines	Covered Lines	Uncovered Lines	Code Lines	Excluded Lines
Entire Solution	69.98%	15,283	10,695	4,588	41,667	27,624
HouseFun.AppBuy.ClientApplication	74.61%	5,475	4,085	1,390	16,276	2,146
App_Start	57.89%	76	44	32	309	0
Controllers	88.70%	1,672	1,483	189	4,429	0
Infrastructure	68.08%	3,628	2,470	1,158	9,714	209
Models	88.89%	99	88	11	1,824	0
HouseFun.AppBuy.Common	86.76%	1,095	950	145	5,537	69
HouseFun.AppBuy.Repository	52.75%	5,519	2,911	2,608	11,564	63
Helper	96.15%	156	150	6	360	0
Repositories	51.48%	5,363	2,761	2,602	11,204	0
HouseFun.AppBuy.Service	86.07%	3,194	2,749	445	8,290	156
Members	92.44%	119	110	9	516	0
Order	100.00%	86	86	0	392	0
Search	91.89%	530	487	43	1,296	0
Services	84.02%	2,459	2,066	393	6,086	0

不同階段的測試歷程

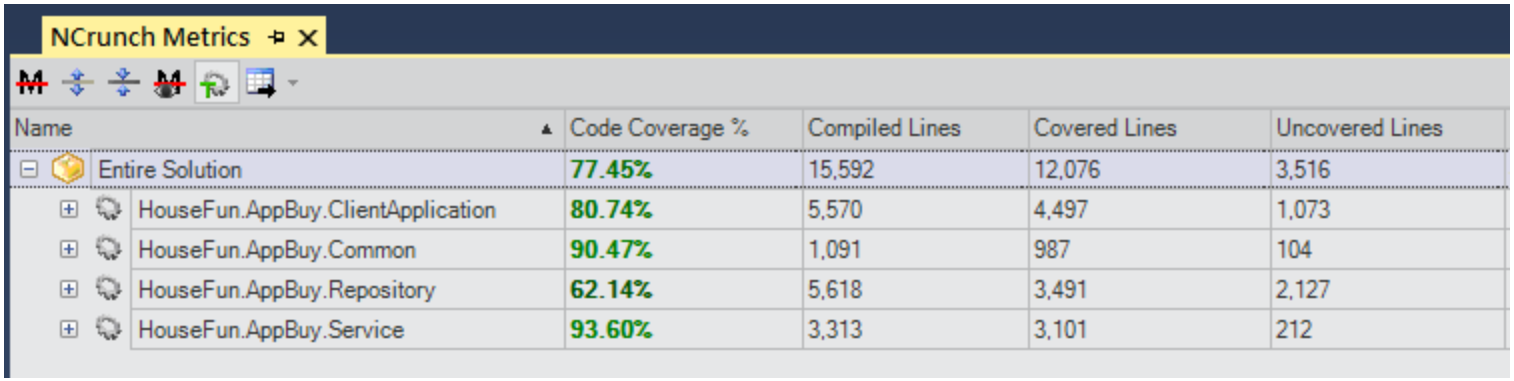
- 2015-11-17



Name	Code Coverage %	Compiled Lines	Covered Lines	Uncovered Lines
Entire Solution	77.41%	15,592	12,070	3,522
.ClientApplication	80.79%	5,570	4,500	1,070
.Common	90.47%	1,091	987	104
.Repository	61.98%	5,618	3,482	2,136
.Service	93.60%	3,313	3,101	212

不同階段的測試歷程

- 2015-11-20



Name	Code Coverage %	Compiled Lines	Covered Lines	Uncovered Lines
Entire Solution	77.45%	15,592	12,076	3,516
HouseFun.AppBuy.ClientApplication	80.74%	5,570	4,497	1,073
HouseFun.AppBuy.Common	90.47%	1,091	987	104
HouseFun.AppBuy.Repository	62.14%	5,618	3,491	2,127
HouseFun.AppBuy.Service	93.60%	3,313	3,101	212

不同階段的測試歷程

- 2015-12-07

NCrunch Metrics

Name	Code Coverage %	Compiled Lines	Covered Lines	Uncovered Lines	Code Lines	Excluded Lines
Entire Solution	77.46%	15,592	12,078	3,514	42,535	29,756
HouseFun.AppBuy.ClientApplication	80.74%	5,570	4,497	1,073	16,835	2,182
HouseFun.AppBuy.Common	90.47%	1,091	987	104	5,504	26
HouseFun.AppBuy.Repository	62.18%	5,618	3,493	2,125	11,751	63
Helper	96.15%	156	150	6	360	0
Repositories	61.20%	5,462	3,343	2,119	11,391	0
HouseFun.AppBuy.Service	93.60%	3,313	3,101	212	8,445	156
Members	92.44%	119	110	9	516	0
Order	100.00%	86	86	0	392	0
Search	91.89%	530	487	43	1,296	0
Services	93.79%	2,578	2,418	160	6,241	0

推動及導入單元測試的時機

- 可以慢慢推廣，但是不強求一定要在專案上執行
- 不要試圖去改他人
- **有推動就會有阻力、有導入就會有抗拒 by 董大偉老師**
- 找尋適當時間點去切入與引導
- 前提：

  必須先找到一個標竿專案，在這個專案裡去實作，

  固定去發佈目前實作的階段

專案的整合測試

- 單元測試是針對程式的最小單元作功能測試，
  主要是關注程式各個流程的執行是否如預期

- 單元測試並無法將不同層面的應用依據步驟給串在一起，
  因為這不是單元測試的本意

專案的整合測試

- 以最為貼近使用者的情境去對系統作各種測試
- 盡量去模擬系統執行的真實情況，以擬真的情境去執行
- 整合測試所在的環境，應該是模擬的測試環境，
  基本上正式環境中該有服務、資源、資料庫等等，
  在測試環境也應有相對應的一份

專案的整合測試

- 一個測試案例只能針對一個情境做測試驗證，無法反應多個情境，但建立多個測試去測試不同的情境而且只有單純的參數不一樣，這樣的做法就有點蠢。
- 這時候就是要用到 SpecFlow 的時候，
  將需求給規格化，建立情境，給予不同的測試案例，
  就可以完成不同參數對同一個測試方法進行驗證。
- **Postman Code Snippet, SpecFlow**

SpecFlow Feature

GeoLocation_Feature.feature | HouseForSearchTest.cs | FavoritesTests.cs

```
Feature: GeoLocation_Feature
    In order to 輸入使用者目前所在位置的經緯度以取得位置資訊
    As a 使用者
    I want to 得到目前位置資訊

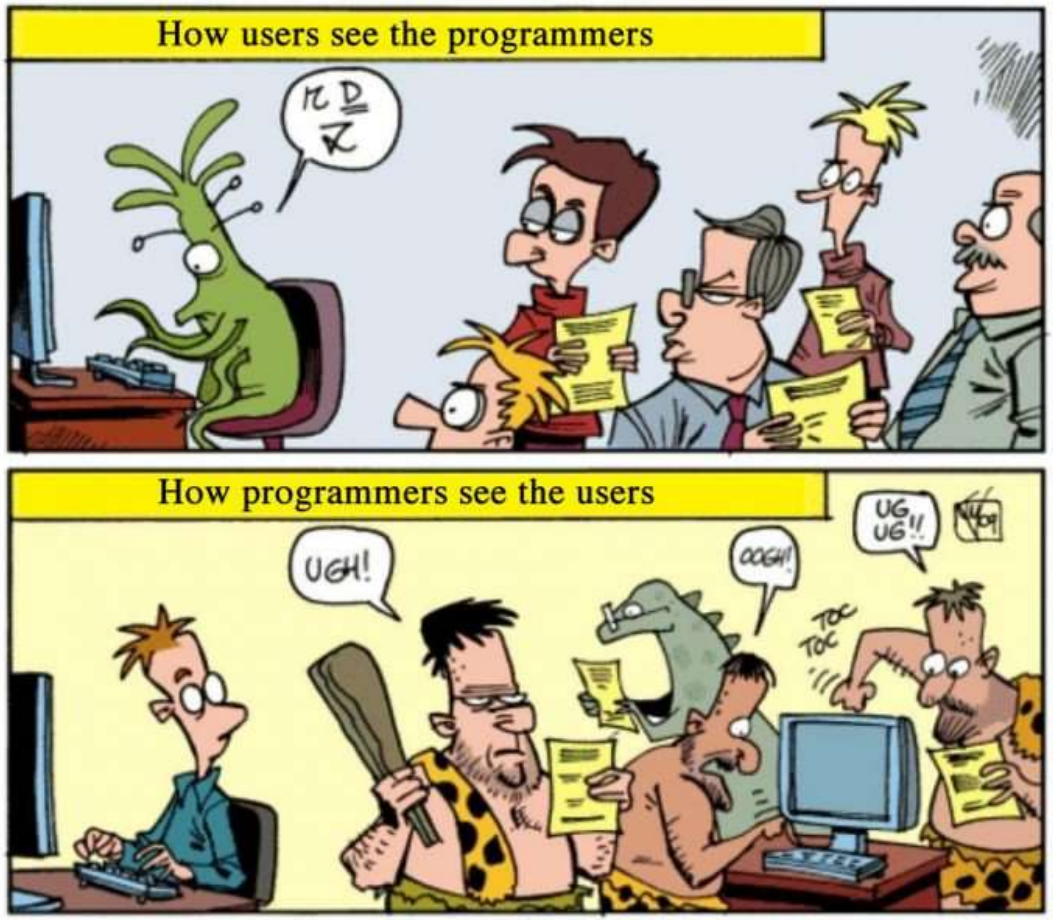
Scenario Outline: 查詢_輸入經緯度
    Given 輸入經度 <Longitude> 與緯度 <Latitude>
    When 呼叫執行Execute方法
    Then 位置資訊的縣市應為 <City> 鄉鎮市區應為 <Area>
    Examples:
    | Latitude          | Longitude          | City | Area |
    | 24.24993164935578 | 120.73521986603737 | 台中市 | 豐原區 |
    | 24.757950812988398 | 121.76636243239045 | 宜蘭縣 | 宜蘭市 |
    | 24.739754618901554 | 121.51820143684743 | 新北市 | 烏來區 |
    | 24.83386468451031 | 121.0162085853517 | 新竹縣 | 竹北市 |
    | 24.230856843363 | 120.5509202600001 | 台中市 | 沙鹿區 |
    | 25.042815207811124 | 121.58565664663911 | 台北市 | 南港區 |
    | 24.924419473051373 | 121.28145525231957 | 桃園市 | 八德區 |
```



demo

好房網買屋APP-Web Api的整合測試

使用 SpecFlow 是為了消弭需求與開發單位的差異



最後需要有東西來呼應需求文件與測試案例



產生文件都是自動化，只需一個動作就可以完成

可以不需開啟 Visual Studio

只需使用 Powershell 執行事先寫好的 Script 指令碼

執行 MSBuild, 跑全部的 SpecFlow 測試, 產生報表…

一氣喝成



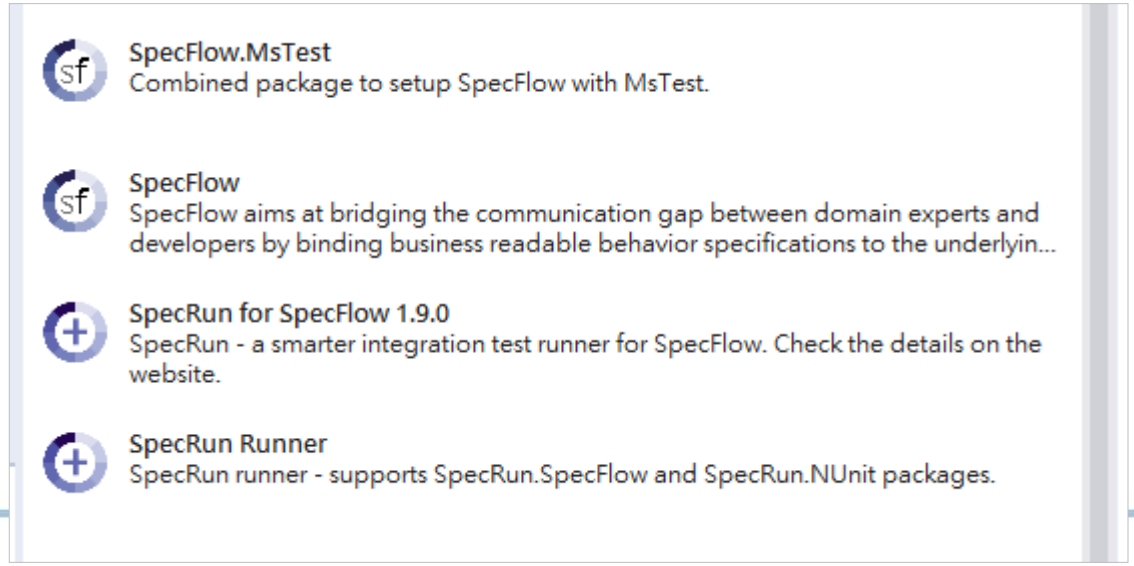
測試專案會需要用到的套件（Nuget 安裝）

SpecFlow

SpecFlow.MsTest （非必要）

SpecRun for SpecFlow 1.9.0

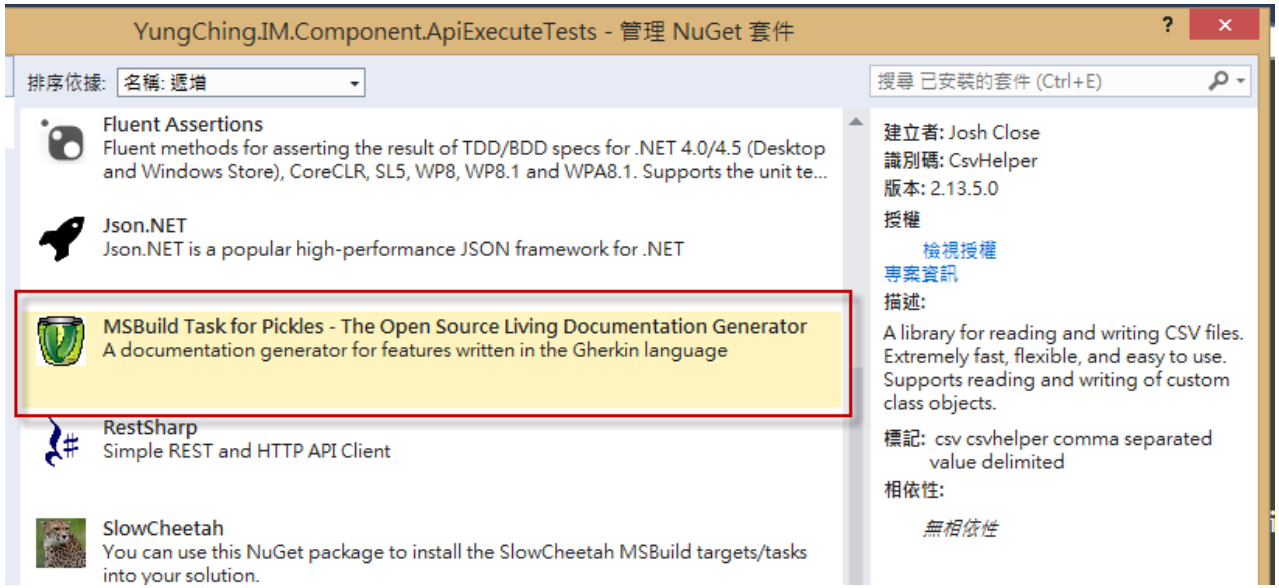
SpecRun Runner （裝了上面就會安裝這一個）



測試專案會需要用到的套件（Nuget 安裝）

MSBuild Task for Pickles

– The Open Source Living Documentation Generator



Living Documentation 這個名字就很銷魂



demo

SpecFlow + SpecRun + Pickles

使用 PowerShell Script 的好處

- 可以不用開啟 Visual Studio 就可以完成專案建置、執行測試、產生報表的一連串工作
- 之後 CI 建置完成後，只要去排定執行指定的 PowerShell Script 檔案就可以完成上述的動作：

  專案建置、執行測試、產生報表

開發心得與感想

專案影響

- 專案結構與開發方式成為公司開發 Web Api 專案的範本
- 依循這個專案的開發模式，例如：

  物件導向、分層架構、單元測試、整合測試
- 制訂一致性的 Api 輸出入規格
- 制訂一致性的 Api 功能：

  Help Page, Swagger,

  Import Collection to Postman

心得與感想

- 溝通，並且敞開心胸
- 發掘問題、面對問題、解決問題、預防問題
- 不要故步自封，找出各種的適合處理方式
- 盡量分享、不要藏私
- 盡力協助、一起面對
- 接受挑戰
- 不要用開發網站程式的心態去開發 APP 的 Web Api 服務

開發時的相關參考書籍和學習資源

ASP.NET Offical WebSite

- Getting Started with ASP.NET Web API 2 (C#)
- Tracing in ASP.NET Web API 2
- HTTP Message Handlers in ASP.NET Web API
- Attribute Routing in ASP.NET Web API 2

twMVC#8

- twMVC#8 – ASP.NET Web API

  by 陳傳興 Bruce

- 簡報檔
- 課後補充資料與專案範例檔



twMVC#14

- twMVC#14 -輕鬆上手ASP.NET Web API 2.1.2

  by 陳傳興 Bruce

- 簡報檔



相關網站

- KingKong Bruce記事 - Web API

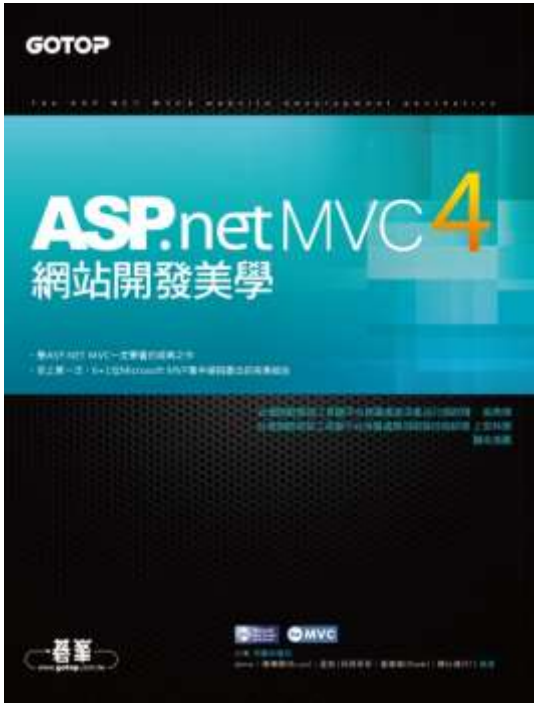


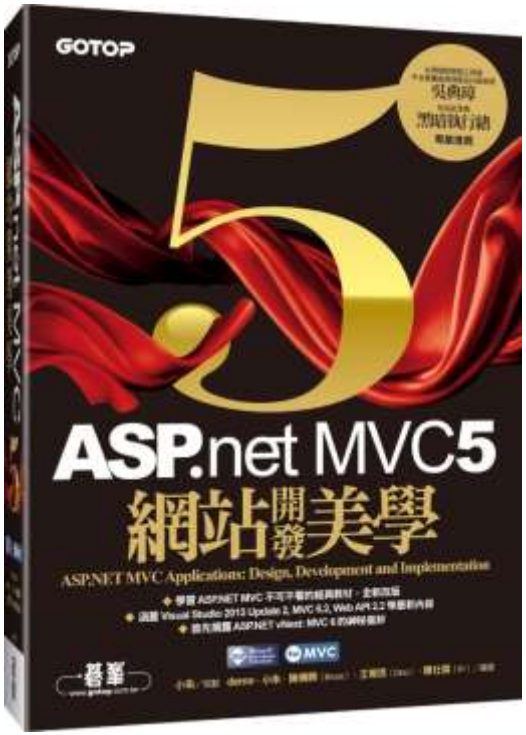
線上學習 – Microsoft Virtual Academy

- Web API Design Jump Start
  http://www.microsoftvirtualacademy.com/training-courses/web-api-design-jump-start
- Web Services 開發實戰: 使用 ASP.NET Web API 2
  http://www.microsoftvirtualacademy.com/training-courses/developing-web-services-using-aspnet-webapi2

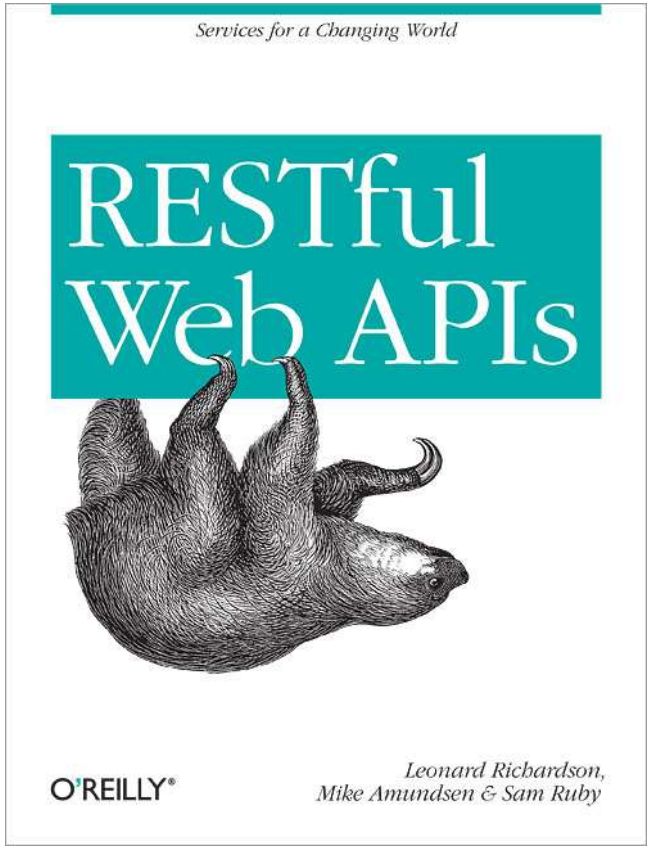
相關書籍

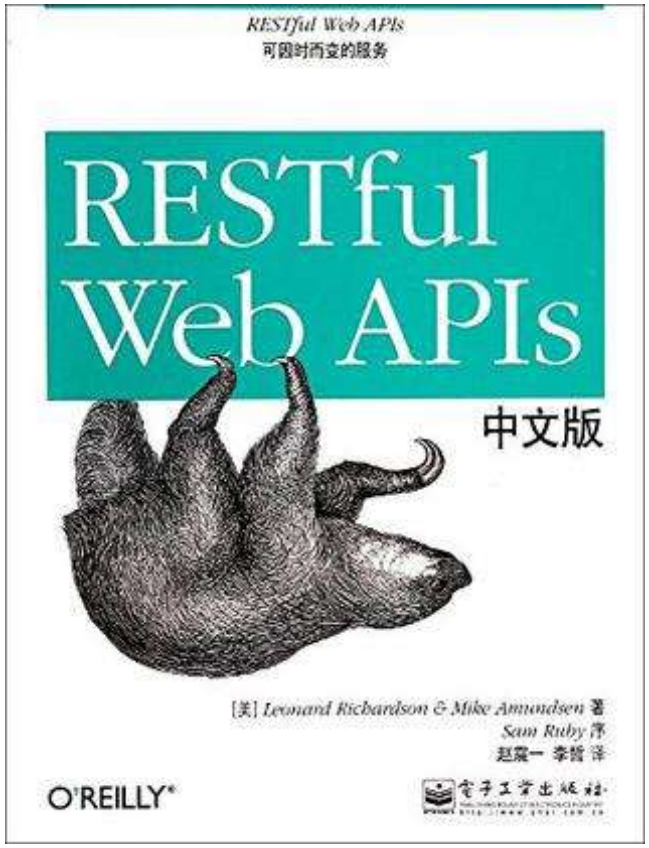
- ASP.NET MVC 4 網站開發美學 (已絕版)
- ASP.NET MVC 5 網站開發美學





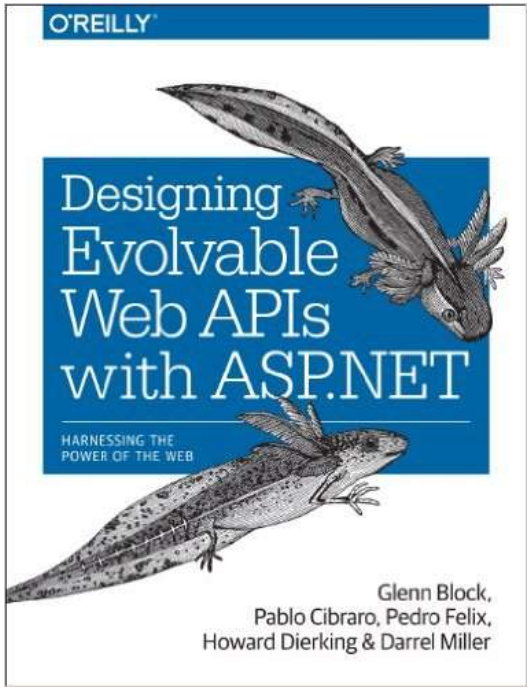
RESTful Web APIs

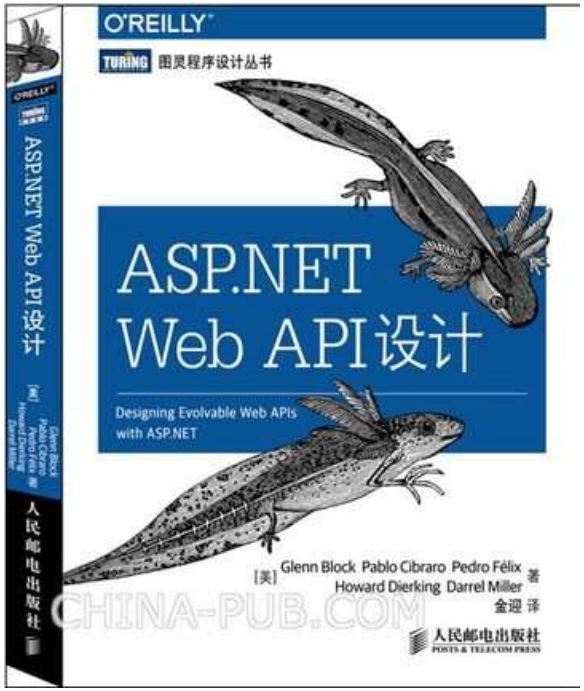




Designing Evolvable Web APIs with ASP.NET

- 英文版:Designing Evolvable Web APIs with ASP.NET
- 簡體中文版:ASP.NET Web API 設計

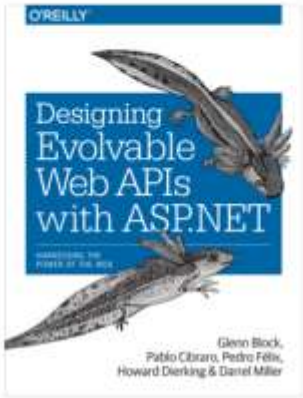




https://github.com/webapibook

Designing Evolvable Web APIs with ASP.NET

O'REILLY Atlas | Titles | Log In / Sign Up | About | Search ...



About the Authors

Glenn Block is a currently working on the Azure team at Microsoft. He was previously a PM on the WCF team working on Microsoft's future HTTP and REST stack as well as a PM on the new Managed Extensibility Framework in .NET 4.0. Glenn has a breadth of experience both inside and outside Microsoft developing software solutions for ISVs and the enterprise.

Pablo Cibraro is a software architect at Tellago Inc and an internationally recognized expert with over 10 years of experience in architecting and implementing large distributed systems with Microsoft Technologies. He has spent the past few years working directly with the Microsoft Patterns & Practices team on sample applications, patterns and guidance for building service-oriented applications with Web Services, Web Services Enhancements (WSE) and Windows Communication Foundation (WCF).

Designing Evolvable Web APIs with ASP.NET

by Glenn Block, Pablo Cibraro, Pedro Felix, Howard Dierking, and Darrel Miller

With this digital Early Release edition of *Programming JavaScript Applications*, you get the entire book bundle in its earliest form—the author's raw and unedited content—so you can take advantage of this content long before the book's official release. You'll also receive updates when significant changes are made, as well as the final ebook version.

Build HTTP services that reach a broad range of clients—including browsers and mobile devices—with ASP.NET Web API. This practical guide shows you how to build evolvable HTTP services using Microsoft's new Web API framework. It included both real world design and technical guidance from members of the ASP.NET Web API team and it's early adopters. It will cover fundamentals of Web API design and how to apply them properly using the technology. You'll learn fundamentals like how to design and select a media type, how to build out your API, and then move on to more advanced topics like how to use content negotiation, leveraging hypermedia, securing and testing your API, and much much more

~~$31.99~~ **Read Online for Free**

Buy it on oreilly.com

http://chimera.labs.oreilly.com/books/1234000001708

Designing Evolvable Web APIs with ASP.NET

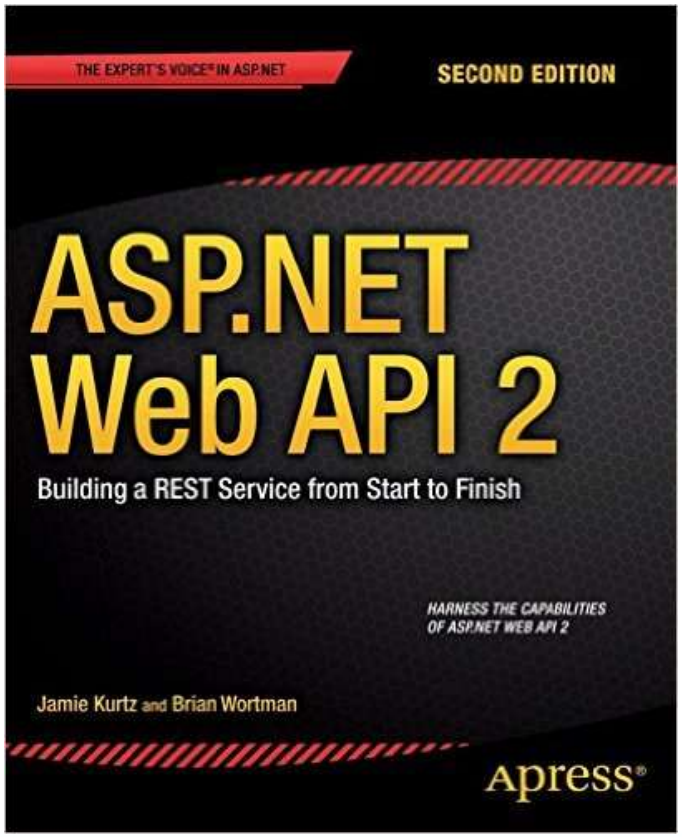


Table of Contents

Foreword
Preface
Why Should You Read This Book?
What Do You Need to Know to Follow Along?
The Hitchhiker's Guide to Navigating This Book
Part I, Fundamentals
Part II, Real-World API Development
Part III, Web API Nuts and Bolts
Conventions Used in This Book

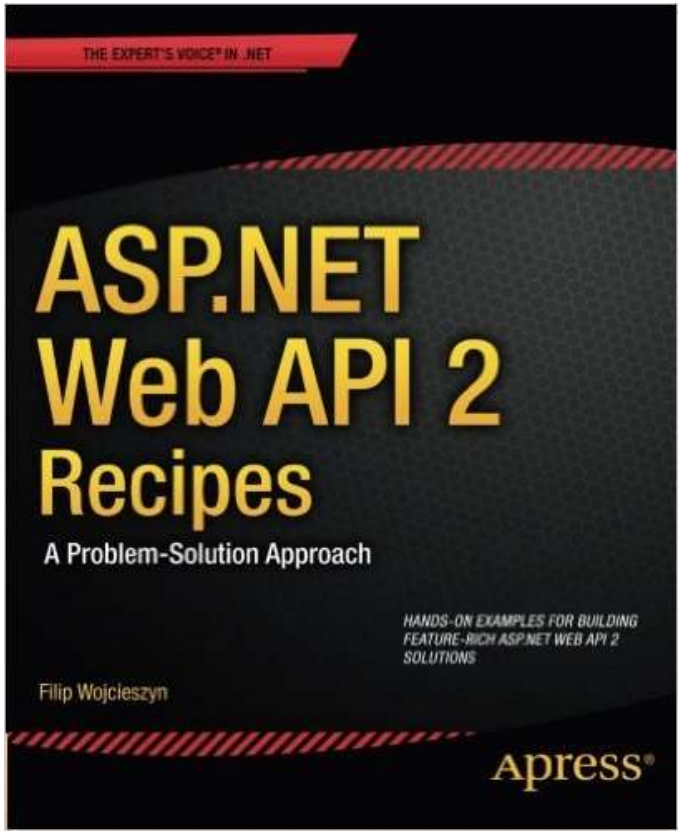
相關書籍

- ASP.NET Web API 2: Building a REST Service from Start to Finish



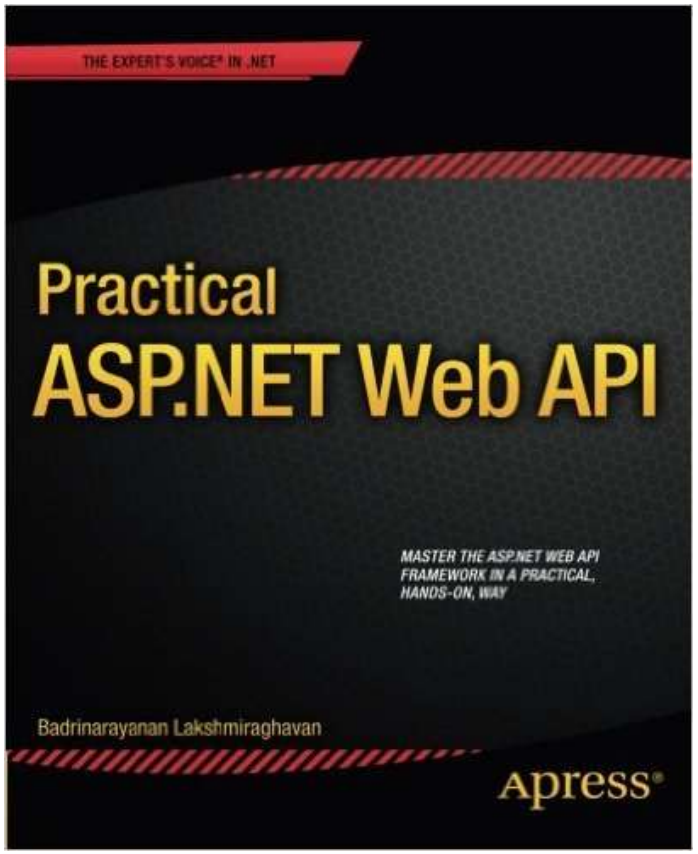
相關書籍

- ASP.NET Web API 2 Recipes: A Problem-Solution Approach



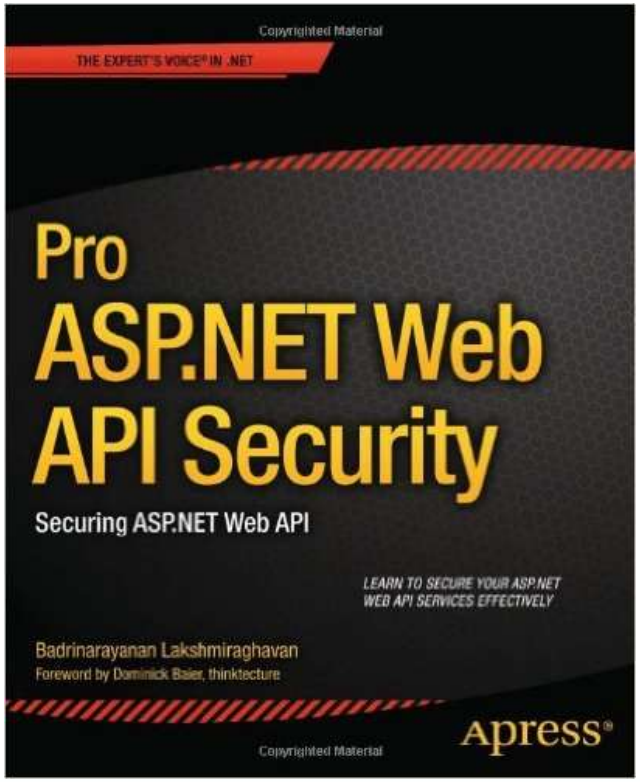
相關書籍

- Practical ASP.NET Web API



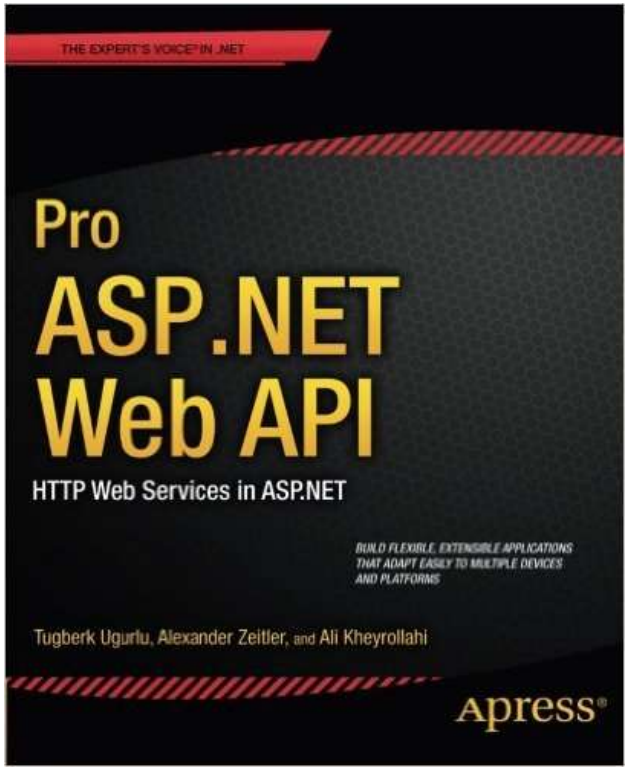
相關書籍

- Pro ASP.NET Web API Security: Securing ASP.NET Web API (Expert's Voice in .NET)



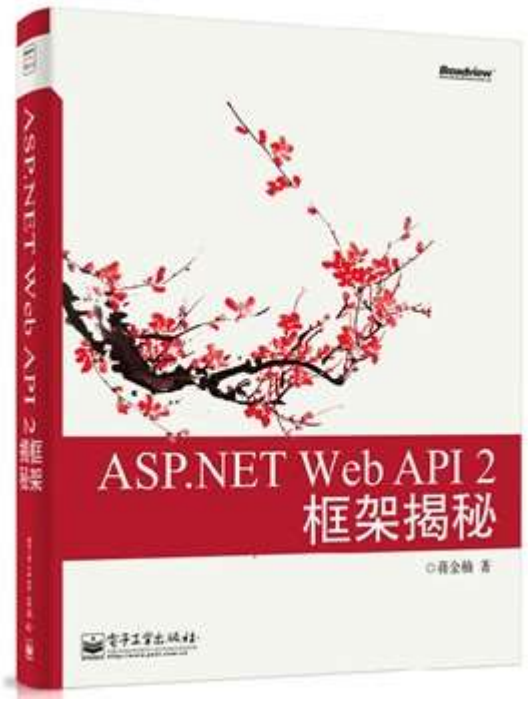
相關書籍

- Pro ASP.NET Web API: HTTP Web Services in ASP.NET (Expert's Voice in .NET)



相關書籍

- ASP.NET Web API 2 框架揭密
- 作者：蔣金楠 （Artech）



http://www.cnblogs.com/artech/p/inside-asp-net-web-api-2-framework.html

Samples @ GitHub

- Samples from ASP.NET Web API 2: Recipes book.
- Example source code that accompanies ASP.NET Web API 2: Building a REST Service from Start to Finish
- Versioning in WebAPI using Route Constraints
- https://github.com/sixeyed/webapi-routeconstraintversioning

Samples @ GitHub

- Example of how to do full-stack unit testing of ASP.NET Web API
- Community Contributions for ASP.NET Web API

課程 – SkillTree.My

- 91 – 自動測試與 TDD 實務開發(使用C#)



相關網站

- In 91 - 點部落



相關網站

- 91 敏捷開發之路 - Facebook



相關網站

- 愛流浪的小風 – 技術隨手寫

  使用 ASP.NET MVC 打造 Web Api 系列 (共30篇)



補充資料

補充資料

mrkt的程式學習筆記-分類或系列文章：

- 分層架構
- AutoMapper
- Dapper
- Elmah
- Nlog
- Visual Studio

補充資料

- mrkt的程式學習筆記:

  Microsoft CodeLens Code Health Indicator

  計算程式碼度量 - Code Metrics Viewer 2013

  使用 CsvHelper - Part.1 資料寫入

  使用 CsvHelper - Part.2 資料讀取

  使用 CsvHelper - Part.3 其他操作說明

補充資料

- [筆記][WebAPI] Message Handler 初體驗-搭配 NLog 記錄所有 WebAPI 的 Request 與 Response | topcat 姍舞之間的極度凝聚 - 點部落

- 從實例學設計模式 by Jace Ju (slides)
  PHP 也有 Day #19 - PHP 返樸歸真系列之從實例學設計模式 by 大澤木小鐵 (Jace Ju)

補充資料

KingKong Bruce記事:

- ASP.NET Web API 文件產生器(1) - Help Page
- ASP.NET Web API 文件產生器(2) - Swagger
- 匯出ASP.NET Web API公開API方法至PostMan Collections

- [ASP.NET] Web API - 匯入API資訊到Postman | No.18 - 點部落

補充資料 - NanoProfiler

Teddy's Knowledge Base - 博客园

NanoProfiler - 适合生产环境的性能监控类库

- 基本功能篇
- 大数据篇
- 实践ELK篇
- https://github.com/englishtown/nanoprofiler

補充資料

[Jenkins] 使用SpecRun產生BDD測試結果報表 | 愛流浪的小風 - 點部落

https://www.dotblogs.com.tw/kirkchen/archive/2013/08/23/use_specrun_to_generate_specflow_testcase_report_and_integrate_with_jenkins.aspx

[料理佳餚] SpecFlow + SpecRun 自動產生測試報表 | 軟體廚房 - 點部落

https://www.dotblogs.com.tw/supershowwei/archive/2015/09/03/153282.aspx

[料理佳餚] SpecFlow + SpecRun + Pickles 讓測試報表變得友善 | 軟體廚房 - 點部落

https://www.dotblogs.com.tw/supershowwei/2015/09/03/153283

補充資料

Pickles 官網

http://docs.picklesdoc.com/en/latest/

Integrating Test Results From SpecRun – Pickles

http://docs.picklesdoc.com/en/latest/IntegratingTestResultsFromSpecRun/

Getting Started | SpecFlow - Additional SpecFlow+ Runner Features

http://www.specflow.org/getting-started/#AdditionalFeatures

補充資料

Channel 9 | PowerShell 入門班(上)

https://channel9.msdn.com/events/Windows-PowerShell-TW/entrylevel/01/player

Channel 9 | PowerShell 入門班(下)

https://channel9.msdn.com/events/Windows-PowerShell-TW/entrylevel/02/player

應該沒什麼時間做 Q&A 了...

史上最強的注音輸入法



千呼萬喚

MAC 平台上，最聰明的注音輸入法
現在立即前往購買 ！

樂在Coding 愛上MVC

完全支援

請放心在 Windows 8 上使用 新自然輸入法 10 系統升級免煩惱，平板模式也沒問題！

Blog 是記錄知識的最佳平台



感謝KKTIX贊助活動報名平台



感謝 KKTIX 贊助 twMVC 活動報名平台

感謝 Jetbrains 贊助贈品



ReSharper

https://www.jetbrains.com/resharper/

感謝 OzCode 贊助贈品



http://www.oz-code.com/

感謝 ALIVE 贊助贈品



https://comealive.io/

業界師資、實戰教學



http://skilltree.my

One more thing…

我們在徵人!

永慶房產集團-資訊處:

- 系統架構師
- 資深軟體開發工程師
- 軟體開發工程師

.NET Framework C# ASP.NET MVC / Web API

謝謝各位

http://mvc.tw

- 本投影片所包含的商標與文字皆屬原著作者所有。
- 本投影片使用的圖片皆從網路搜尋。
- 本著作係採用姓名標示-非商業性-相同方式分享 3.0 台灣授權。閱讀本授權條款，請到 http://creativecommons.org/licenses/by-nc-sa/3.0/tw/，或寫信至Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.