# 精通.NET MVC的本質與開發架構

* 1.強化C#物件導向設計基礎觀念(C# OOP Basic Concept(封裝、繼承、多型))
* 2.說明甚麼是.NET Framework的BCL和CLR? 甚麼叫擴充函式庫?
* 3.事件Event驅動和Delegate(委派)的最主要用途是甚麼?
* 4.詳解LinQ技術、Lambda語法及語法樹Expression Tree觀念
* 5.進階物件導向設計(深入探討IoC、DI、AOP)
* 6.MVC Design Pattern是甚麼? 你現在用的是哪一種設計模式? 甚麼是反模式?
* 7.Web API
* 8.Entity Framework

1. 強化C#物件導向設計基礎觀念(C# OOP Basic Concept(封裝、繼承、多型))
   1. [型別、變數和值](https://docs.microsoft.com/zh-tw/dotnet/csharp/programming-guide/types/#types-variables-and-values)
   2. 封裝、[繼承](https://docs.microsoft.com/zh-tw/dotnet/csharp/programming-guide/classes-and-structs/inheritance)、[多型](https://docs.microsoft.com/zh-tw/dotnet/csharp/programming-guide/classes-and-structs/polymorphism)
   3. [C# 關鍵字](https://docs.microsoft.com/zh-tw/dotnet/csharp/language-reference/keywords/)
   4. 微軟官方教學文件：
      1. Microsoft C# 指南 <https://docs.microsoft.com/zh-tw/dotnet/csharp/>
      2. C# 語言參考：<https://docs.microsoft.com/zh-tw/dotnet/csharp/language-reference/>
2. 說明甚麼是.NET Framework的BCL和CLR? 甚麼叫擴充函式庫?
3. [.NET Framework概觀](https://www.mpinfo.com.tw/TechnologyColumnFiles/PB_T_201006.pdf)
   1. CLR執行環境
   2. BCL是.NET Framework[基礎類別庫](https://zh.wikipedia.org/wiki/%E5%9F%BA%E7%A1%80%E7%B1%BB%E5%BA%93)
   3. 透過C#擴展方法使您可以向現有類型添加方法，而無需創建新的派生類型，重新編譯或以其他方式修改原始類型。 C#擴展方法是一種特殊的靜態方法，它被稱為擴展類型的實例方法。

微軟官方教學文件及維基百科：

[組件資訊清單 Assembly Manifest](https://docs.microsoft.com/zh-tw/dotnet/framework/app-domains/assembly-manifest)

[可移植執行體 PE](https://zh.wikipedia.org/wiki/%E5%8F%AF%E7%A7%BB%E6%A4%8D%E5%8F%AF%E6%89%A7%E8%A1%8C)

[通用語言運行環境 CLR](https://zh.wikipedia.org/wiki/%E9%80%9A%E7%94%A8%E8%AA%9E%E8%A8%80%E9%81%8B%E8%A1%8C%E5%BA%AB)、[(CLR) 概觀](https://docs.microsoft.com/zh-tw/dotnet/standard/clr)

[通用中間語言CIL/MSIL](https://zh.wikipedia.org/wiki/%E9%80%9A%E7%94%A8%E4%B8%AD%E9%97%B4%E8%AF%AD%E8%A8%80)

[即時編譯JIT(Just-in-time compilation)](https://zh.wikipedia.org/wiki/%E5%8D%B3%E6%99%82%E7%B7%A8%E8%AD%AF)

[應用程式定義域 Application Domain](https://docs.microsoft.com/zh-tw/dotnet/framework/app-domains/application-domains)、[使用應用程式定義域](https://docs.microsoft.com/zh-tw/dotnet/framework/app-domains/use)

[擴充方法 (C# 程式設計手冊)](https://docs.microsoft.com/zh-tw/dotnet/csharp/programming-guide/classes-and-structs/extension-methods)

1. 事件Event驅動和Delegate(委派)的最主要用途是甚麼?
   1. [Delegate](https://docs.microsoft.com/zh-tw/dotnet/csharp/delegates-overview)
   2. [Event](https://docs.microsoft.com/zh-tw/dotnet/standard/events/how-to-raise-and-consume-events)

參考資料：<https://eric0806.blogspot.com/2015/01/dotnet-delegate-usage.html>

1. 詳解LinQ技術、Lambda語法及語法樹Expression Tree觀念
   1. [Lambda](https://docs.microsoft.com/zh-tw/dotnet/standard/delegates-lambdas)
   2. [泛型類型](https://docs.microsoft.com/zh-tw/dotnet/standard/generics)
   3. [LinQ](https://docs.microsoft.com/zh-tw/dotnet/csharp/linq/) 、[linq](https://docs.microsoft.com/zh-tw/dotnet/standard/using-linq)
   4. [Expression Tree](https://docs.microsoft.com/zh-tw/dotnet/csharp/programming-guide/concepts/expression-trees/)
2. 進階物件導向設計(深入探討IoC、DI、AOP)
   1. 名詞介紹：DIP依賴倒置原則(Dependency-Inversion Principle)、  
      IoC控制反轉(Inversion of Control)、DI依賴注入(Dependency Injection)
      1. [深入理解DIP、IoC、DI以及IoC容器](https://www.cnblogs.com/liuhaorain/p/3747470.html#title_2)
   2. AOP
      1. [術語與觀念](https://openhome.cc/Gossip/SpringGossip/AOPConcept.html)
      2. [AOP維基](https://zh.wikipedia.org/wiki/%E9%9D%A2%E5%90%91%E5%88%87%E9%9D%A2%E7%9A%84%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1)
      3. [利用C#實現AOP常見的幾種方法詳解](https://www.cnblogs.com/panpanwelcome/p/8617552.html)(靜態、動態、3種方式)
      4. 介紹.NET Framework提供的兩個AOP類別： [RealProxy](https://msdn.microsoft.com/zh-cn/library/dn574804.aspx)、[MarshalByRefObject](https://ithelp.ithome.com.tw/articles/10197872)(延伸[ContextBoundObject](https://docs.microsoft.com/zh-tw/dotnet/api/system.contextboundobject?view=netframework-4.8)、[其他參考1](https://dotblogs.com.tw/steventsai/2017/06/08/143849)、 [2](https://fecbob.pixnet.net/blog/post/39293239))
3. MVC Design Pattern是甚麼? 你現在用的是哪一種設計模式? 甚麼是反模式?
   1. 談到.NET MVC不如先聊聊[Architectural Pattern](https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013)(軟體架構模式/模型)觀念
   2. Design Pattern(軟體設計模式)觀念：  
      [重要觀念(念他的前兩段)](https://www.itread01.com/p/369053.html)、[維基百科](https://zh.wikipedia.org/wiki/%E8%AE%BE%E8%AE%A1%E6%A8%A1%E5%BC%8F_(%E8%AE%A1%E7%AE%97%E6%9C%BA))
   3. 關於各種GoF設計模式的介紹文章：[GoF 的23種DP](https://blog.csdn.net/zhangcanyan/article/details/51622380)、[相關文章1](https://ithelp.ithome.com.tw/articles/10201706)、
   4. 若當今物件導向程式設計奉GoF Design Pattern為圭臬，則[SOLID](https://zh.wikipedia.org/wiki/SOLID_(%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1%E8%AE%BE%E8%AE%A1))原則就是設計品質的度量衡工具
   5. 什麼是反模式？  
      Design pattern 告訴我們軟體工程中解決問題的正確方法，而AntiPattern則是告訴我們錯誤的方法。  
      [維基百科](https://zh.wikipedia.org/wiki/%E5%8F%8D%E9%9D%A2%E6%A8%A1%E5%BC%8F)  
      其他例子：[軟體測試的反模式](https://medium.com/m/signin)、
4. WebAPI
   1. [理解WebAPI的 HttpMessageHandler](https://docs.microsoft.com/zh-tw/aspnet/web-api/overview/advanced/http-message-handlers)