ASP.NET CORE 新手上路的七個路坑

ASP.NET開發者進入ASP.NET CORE開發需注意的N個面向

同樣都是C#開發WEB後端，哪些事情不一樣了？

0) **.NET Standard** 和 **.NET Core** 的差異
1) Cross Platform Dev Toolchain
2) DI/IoC & middleware 架構
3) 設定檔與 Options Pattern
4) EFCore code-first / DB provider for No-SQL
5) ASP.NET Core Identity
6) Unit/Integration Test utility
7) Deployment & 容器化(Dockerize)

.NET STANDARD 和 .NET CORE 的差異

- ✓ **.NET Core** 和 **.NET Standard** 是不一樣的東西!!!
- ✓ 由**.NET Standard**專案所產生的class lib可被**.NET Core**專案使用，反之不行。

.NET Standard

- 一種API標準，由 .NET基金會訂立，讓C#產生的組件可以在各類可用C#開發應用程式的平台上引用。

.NET Core

- 微軟根據.NET Standard的標準API，實作出的一種跨平台執行環境。

連第三方廠商Unity3D 都跟你提醒：
.NET Standard和.NET Core不一樣的

https://docs.unity3d.com/Manual/dotnetProfileSupport.html

Managed plugins

Managed code plugins compiled outside of Unity can work with either the .NET Standard 2.0 profile or the .NET 4.x profile in Unity. The following table describes configurations Unity supports:

		API Compatibility Level:	
		.NET Standard 2.0	**.NET 4.x**
Managed plugin compiled against:			
	.NET Standard	Supported	Supported
	.NET Framework	Limited	Supported
	.NET Core	Not Supported	Not Supported

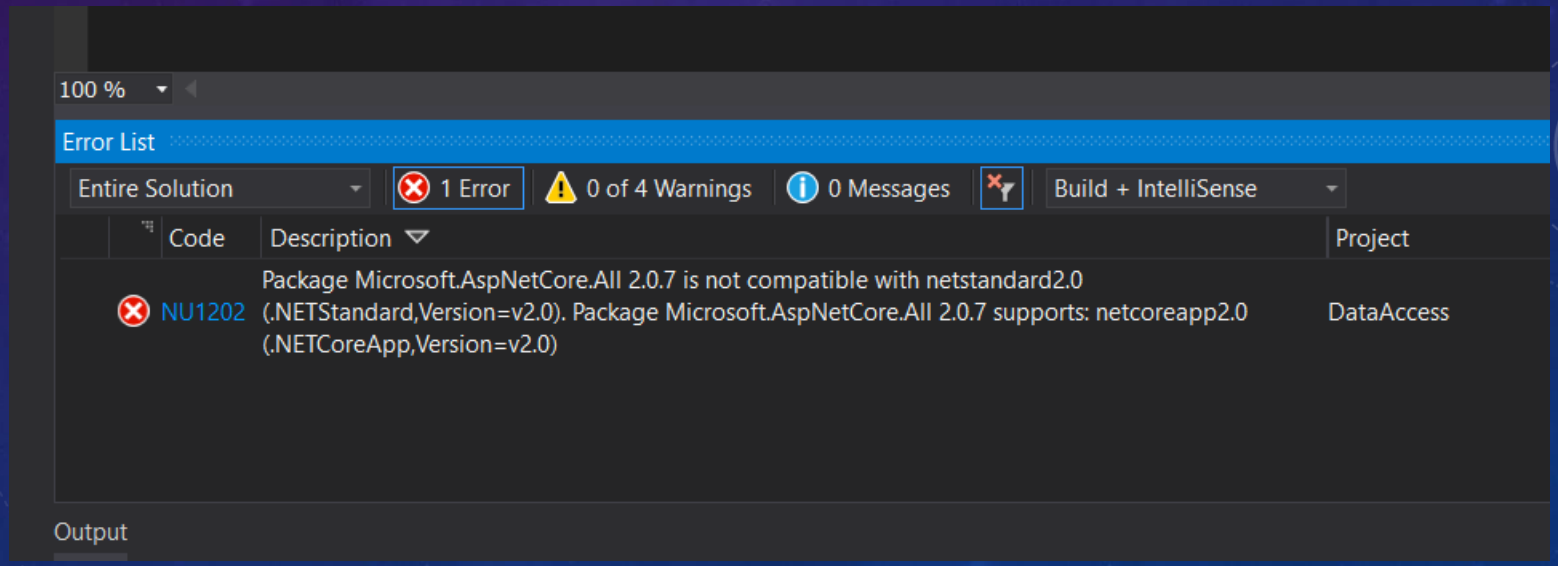
Note:

- Managed plugins compiled against any version of the .NET Standard work with Unity.
- Limited support indicates that Unity supports the configuration if all APIs used from the .NET Framework are present in the .NET Standard 2.0 profile. However, the .NET Framework API is a superset of the .NET Standard 2.0 profile, so some APIs are not available.

.NET STANDARD 和 .NET CORE 的差異

由**.NET Standard**專案所產生的class lib可被**.NET Core**專案使用，反之不行

- 可用來做前後端專案的程式碼/DTO類別共用。
- 配合新的csproj專案格式可同時產出給不同C#平台的組件。
  (不過仍要注意不同平台所支援的C#版本，例如Unity所用的mono compiler，目前支援的C# 7.x語法仍未齊全 http://github.com/mono/mono/issues/6854)



Visual Studio 2017開新 .NET Standard專案

New Project

- Recent
- Installed
  - Visual C#
    - Windows Universal
    - Windows Classic Desktop
    - Web
    - Office/SharePoint
    - .NET Core
    - .NET Standard
    - Android
    - Cloud
    - Cross-Platform
    - Extensibility
    - iOS
    - Prism
    - Test
    - tvOS
    - WCF
    - Workflow

Not finding what you are looking for?

Open Visual Studio Installer

Sort by: Default

Search (Ctrl+E)

Class Library (.NET Standard) Visual C#

Class Library (.NET Standard)

Type: Visual C#

A project for creating a class library that targets .NET Standard.

Name: NetStandardClassLibrary

Location: D:\myTemp\ClassLibrary1 Browse...

Solution: Add to solution

Solution name: NetStandardClassLibrary

☑ Create directory for solution

☑ Create new Git repository

OK Cancel

Visual Studio 2017開新 .NET Core Class library專案

New Project

- Recent
- Installed
  - Visual C#
    - Windows Universal
    - Windows Classic Desktop
    - Web
    - Office/SharePoint
    - .NET Core
    - .NET Standard
    - Android
    - Cloud
    - Cross-Platform
    - Extensibility
    - iOS
    - Prism
    - Test
    - tvOS
    - WCF
    - Workflow
  - Visual Basic

Not finding what you are looking for?

Open Visual Studio Installer

Sort by: Default

Search (Ctrl+E)

Console App (.NET Core)	Visual C#
Class Library (.NET Core)	Visual C#
MSTest Test Project (.NET Core)	Visual C#
xUnit Test Project (.NET Core)	Visual C#
ASP.NET Core Web Application	Visual C#

Type: Visual C#

A project for creating a class library that targets .NET Core.

Name: ClassLibrary1

Location: D:\repo

Solution name: ClassLibrary1

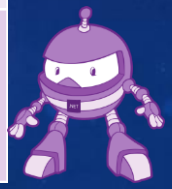
Browse...

☑ Create directory for solution

☑ Create new Git repository

OK Cancel

.NET	執行環境	是否開源	使用場合
.NET Framework .NET Native Windows 10 UWP	Windows XBox HoloLens	No (但有reference source https://referencesource.microsoft.com/)	傳統.NET Framework程式：WinForm, WPF, ASP.NET, WCF Windows 10 UWP平台app
.NET Core	Desktop OS • Windows • MacOS • Linux	Yes https://github.com/dotnet/core	• .NET Standard的一種範例實作平台 • 建立跨平台命令列程式/微服務/網頁App(ex: ASP.NET Core)
Xamarin (mono runtime)	Windows MacOS Linux iOS Android UWP Tizen	Yes https://github.com/mono/mono	建立跨平台手機/桌面應用程式
Unity3D (mono + IL2Cpp) http://unity3d.com/	Windows MacOS iOS Android UWP WebGL 家用電玩主機	No (IL2Cpp沒開源)	建立2D/3D應用程式或遊戲的遊戲引擎 (2018.1的版本開始支援.NET Standard，先前的相容.NET Framework v3.5 & v4.6兩種版本的組件)
.NET Standard	n/a	Yes https://github.com/dotnet/standard	提供上述所有可使用C#開發的平台一種能彼此相容可叫用的API標準，專案格式能用來建立這些平台共用的函式庫組件



.NET STANDARD 和 .NET CORE 的差異

- ✓ 不清楚.NET Core基本指令和專案檔操作的，建議上Udemy線上課程：

給 C# 開發人員的第一堂 .NET Core 入門課
https://www.udemy.com/netcore2/

CROSS PLATFORM DEV TOOLCHAIN

- ✓ **一堆原本在IDE上的操作動作絕對都會有指令列CLI可以用。**
- ✓ **.NET Core SDK** *Windows/MacOS/Linux*跨平台的，配合VSCode就處處可免費開工。
- ✓ 用 ASP.NET Core 2.1 開發建議使用 “Microsoft.AspNetCore.App” 這個meta package，而不是原本的 “Microsoft.AspNetCore.All”。
- ✓ 名稱不要含『.』，否則會造成產生程式碼的命名空間(namespace)路徑錯誤。

.NET Core SDK

- Windows/MacOS/Linux上都可以安裝執行的CLI程式，且可安裝多個SDK版本。(使用global.json 來切換專案上要使用的SDK)
- 舊版/預覽版SDK的各平台安裝說明：http://github.com/dotnet/core/tree/master/release-notes/download-archives

- ✓ .NET Core 2.1之後使用安裝全域指令(https://aka.ms/global-tools)的nuget套件，取代先前專案檔內的<DotNetCliToolReference>設定。
- ASP.NET Core 的全域指令工具： http://github.com/aspnet/DotNetTools/

.NET

dotnet-aspnet-codegenerator 2.1.0-rc1-final

Code Generation tool for ASP.NET Core. Contains the dotnet-aspnet-codegenerator command used for generating controllers and views.

ⓘ This is a prerelease version of dotnet-aspnet-codegenerator.

.NET CLI

```
> dotnet tool install --global dotnet-aspnet-codegenerator --version 2.1.0-rc1-final
```

ⓘ This package contains a .NET Core Global Tool you can call from the shell/command line.

Dependencies

This package has no dependencies.

ASP.NET Core code generator:
http://www.nuget.org/packages/dotnet-aspnet-codegenerator

```
PS D:\myTemp\aspnetcoreDemo\MvcMovie> dotnet aspnet-codegenerator -h


Usage: aspnet-codegenerator [arguments] [options]

Arguments:
  generator  Name of the generator. Check available generators below.

Options:
  -p|--project             Path to .csproj file in the project.
  -n|--nuget-package-dir
  -c|--configuration       Configuration for the project (Possible values: Debug/ Release)
  -tfm|--target-framework  Target Framework to use. (Short folder name of the tfm. eg. net46)
  -b|--build-base-path
  --no-build

Available generators:
  area      : Generates an MVC Area.
  controller: Generates a controller.
  identity  : Generates an MVC Area with controllers and
  razorpage : Generates RazorPage(s).
  view      : Generates a view.


RunTime 00:00:02.50
```

CROSS PLATFORM DEV TOOLCHAIN

.NET Core SDK

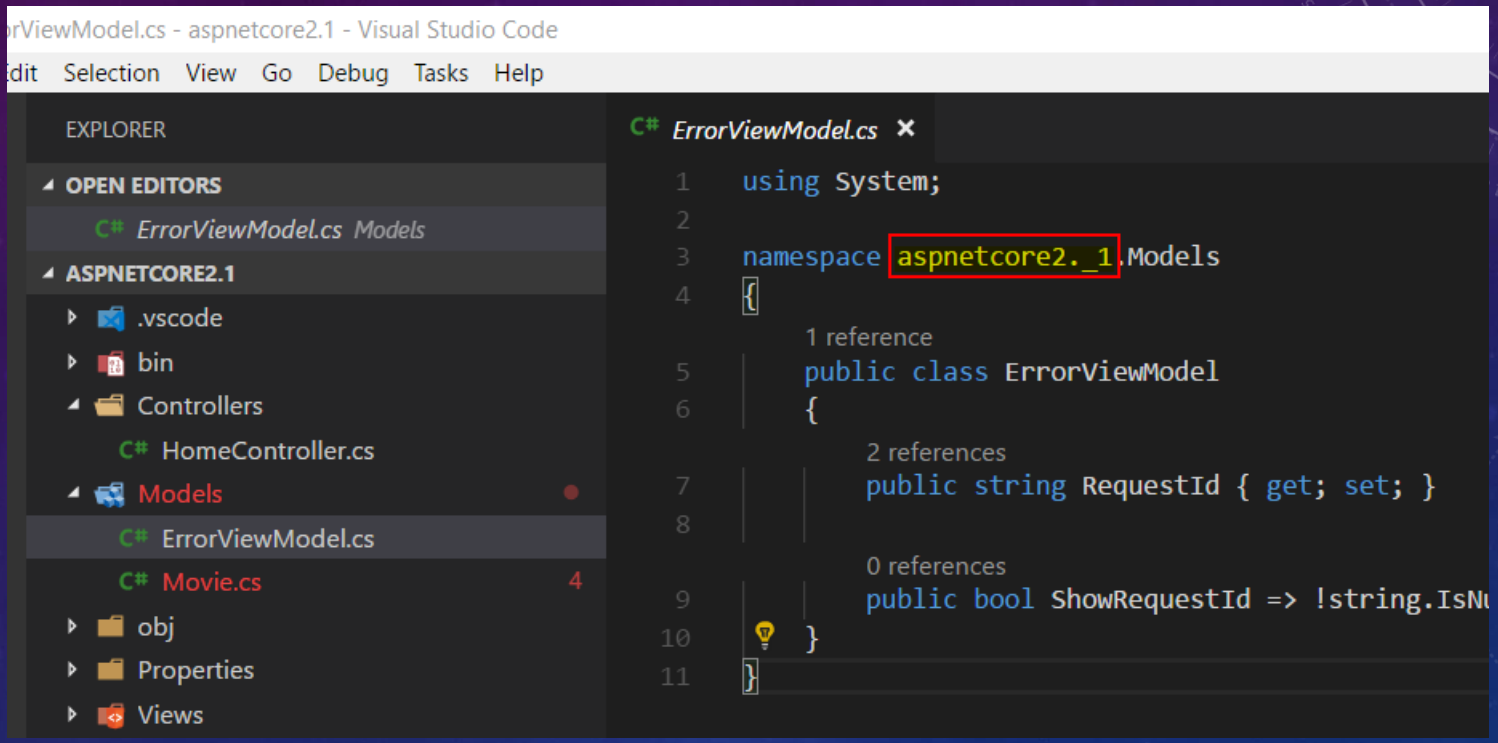
專案範本可自行安裝客製版(template pack)：

- ✓ 官方推薦的專案範本：
  http://github.com/dotnet/templating/wiki/Available-templates-for-dotnet-new

建立新專案的CLI指令：

```
dotnet new [project_template_name] -n [目錄兼專案檔名稱]
```

專案名稱不要含『.』，會造成產生程式碼的命名空間路徑錯誤。



在Linux上安裝NUnit 3 (https://github.com/nunit/dotnet-new-nunit)
的template pack示範：

```
vmusr@vm-ubuntu1804:~$ dotnet new -i NUnit3.DotNetNew.Template
Getting ready...
  Restoring packages for /home/vmusr/.templateengine/dotnetcli/v2.1.300-preview2-008533/scratch/restore.csproj...
  Installing NUnit3.DotNetNew.Template 1.4.0.
  Generating MSBuild file /home/vmusr/.templateengine/dotnetcli/v2.1.300-preview2-008533/scratch/obj/restore.csproj.nuget.g.props.
  Generating MSBuild file /home/vmusr/.templateengine/dotnetcli/v2.1.300-preview2-008533/scratch/obj/restore.csproj.nuget.g.targets.
  Restore completed in 3.87 sec for /home/vmusr/.templateengine/dotnetcli/v2.1.300-preview2-008533/scratch/restore.csproj.

Usage: new [options]

Options:
  -h, --help          Displays help for this command.
  -l, --list          Lists templates containing the specified name. If no name is specified, lists all templates.
  -n, --name          The name for the output being created. If no name is specified, the name of the current directory is used.
  -o, --output        Location to place the generated output.
  -i, --install       Installs a source or a template pack.
  -u, --uninstall     Uninstalls a source or a template pack.
  --nuget-source      Specifies a NuGet source to use during install.
  --type              Filters templates based on available types. Predefined values are "project", "item" or "other".
  --force             Forces content to be generated even if it would change existing files.
  -lang, --language   Filters templates based on language and specifies the language of the template to create.


Templates                                        Short Name       Language          Tags
----------------------------------------------------------------------------------------------------
Console Application                              console          [C#], F#, VB      Common/Console
Class library                                    classlib         [C#], F#, VB      Common/Library
Unit Test Project                                mstest           [C#], F#, VB      Test/MSTest
NUnit 3 Test Project                             nunit            [C#], F#, VB      Test/NUnit
xUnit Test Project                               xunit            [C#], F#, VB      Test/xUnit
Razor Page                                       page             [C#]              Web/ASP.NET
MVC ViewImports                                  viewimports      [C#]              Web/ASP.NET
MVC ViewStart                                    viewstart        [C#]              Web/ASP.NET
ASP.NET Core Empty                               web              [C#], F#          Web/Empty
ASP.NET Core Web App (Model-View-Controller)     mvc              [C#], F#          Web/MVC
ASP.NET Core Web App                             razor            [C#]              Web/MVC/Razor Pages
ASP.NET Core with Angular                        angular          [C#]              Web/MVC/SPA
ASP.NET Core with React.js                       react            [C#]              Web/MVC/SPA
```

CROSS PLATFORM DEV TOOLCHAIN

.NET Core SDK

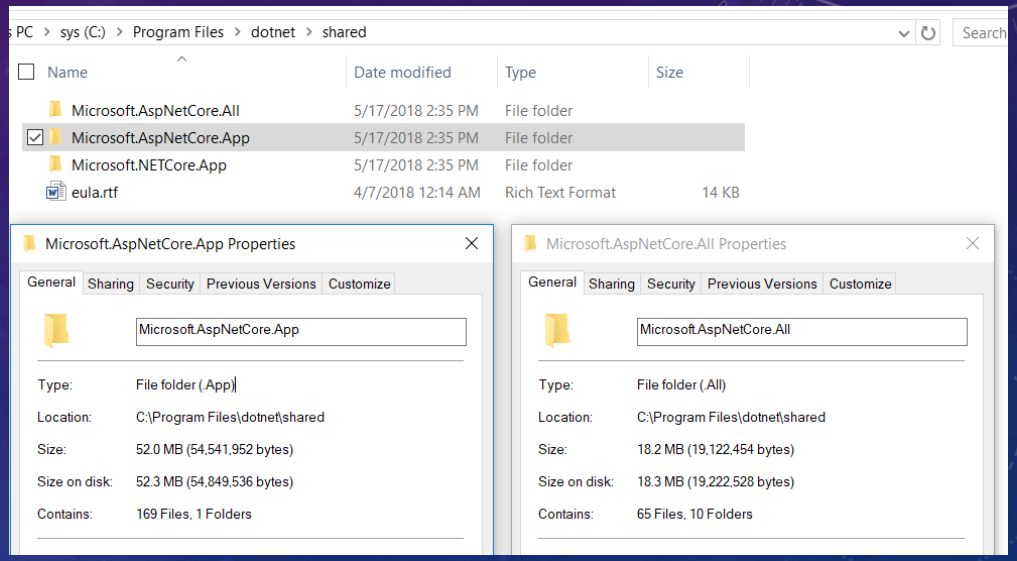
- 可產生 **.NET Core & .NET Standard** 的class lib專案
- 用 `dotnet new classlib -h` 來看有哪些“Framework” (-f參數)是用來指定建立的是 **.NET Core** 還是 **.NET Standard**的專案

Command Prompt

```
[redacted]>dotnet new classlib -h
Usage: new [options]

Options:
  -h, --help          Displays help for this command.
  -l, --list          Lists templates containing the specified name. If no name is specified,
  -n, --name          The name for the output being created. If no name is specified, the name
  -o, --output        Location to place the generated output.
  -i, --install       Installs a source or a template pack.
  -u, --uninstall     Uninstalls a source or a template pack.
  --nuget-source      Specifies a NuGet source to use during install.
  --type              Filters templates based on available types. Predefined values are "proje
  --force             Forces content to be generated even if it would change existing files.
  -lang, --language   Filters templates based on language and specifies the language of the te


Class library (C#)
Author: Microsoft
Description: A project for creating a class library that targets .NET Standard or .NET Core
Options:
  -f|--framework  The target framework for the project.
                      netcoreapp2.1     - Target netcoreapp2.1
                      netstandard2.0    - Target netstandard2.0
                  Default: netstandard2.0

  --no-restore    If specified, skips the automatic restore of the project on create.
                  bool - Optional
                  Default: false / (*) true


* Indicates the value used if the switch is provided without a value.

d:\myTemp\netcorelib>
```

CROSS PLATFORM DEV TOOLCHAIN

- 用 ASP.NET Core 2.1 開發新專案，建議使用 **“Microsoft.AspNetCore.App”** 這個 meta package，而不是原本的 **“Microsoft.AspNetCore.All”**，以免會有非微軟支援的相依第三方 nuget套件版本更新問題：https://github.com/aspnet/Announcements/issues/287



CROSS PLATFORM DEV TOOLCHAIN

Visual Studio,
Visual Studio for Mac,
VSCode (Visual Studio Code),
JetBrains Rider

- 底層其實也是呼叫SDK來執行各種動作。
- Visual Studio在專案使用的SDK升版以及相關nuget套件配合升級時，建議在升版時自己手動改.csproj裡的<PackageReference>，用Visual Studio的Nuget介面操作慢，而且很容易失敗或整個Visual Studio當掉。
- 所使用的SDK版本會讓Visual Studio的Nuget管理介面預設鎖住某些框架用的nuget套件的版本不能更新，如果要更新，得自己手動改.csproj的內容。

Class1.cs | NuGet: ClassLibrary1

Browse | Installed | Updates

NuGet Package Manager: ClassLibrary1

Search (Ctrl+E) | Include prerelease | Package source: nuget.org

Microsoft.NETCore.App by Microsoft — v2.0.0

A set of .NET API's that are included in the default .NET Core application model.

Microsoft.NETCore.App

Installed: 2.0.0 | Uninstall

Version: 2.0.0 | Update

2.0.0

Blocked by project

. To update the
ich it belongs. Learn

Following versions are unavailable due to additional constraints in the project or packages.config

2.0.7
2.0.6
2.0.5
2.0.4
2.0.3
1.1.8
1.1.7

ıded in the
ıodel.
l04c98b69f28d
age requires at

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'ClassLibrary1' (1 project)
- ClassLibrary1
  - Dependencies
  - Class1.cs

Solution Explorer | Team Explorer

Properties

✓ 在專案檔內加上**<RuntimeFrameworkVersion>**強制更版

AspNetCoreFile...leTable.csproj

```
1 <Project Sdk="Microsoft.NET.Sdk.Web">
2   <PropertyGroup>
3     <TargetFramework>netcoreapp2.0</TargetFramework>
4     <RuntimeFrameworkVersion>2.0.7</RuntimeFrameworkVersion>
5   </PropertyGroup>
6   <ItemGroup>
```

CROSS PLATFORM DEV TOOLCHAIN

VSCode (Visual Studio Code) 作為IDE，相關的擴充套件：

- .NET Core Extension Pack
  https://marketplace.visualstudio.com/items?itemName=doggy8088.netcore-extension-pack
- C# for Visual Studio Code
  https://marketplace.visualstudio.com/items?itemName=ms-vscode.csharp
  - ✓ 通常在安裝新套件之後的找不到ooxx編譯錯誤可由reload VSCode Window解決
- C# Extensions https://marketplace.visualstudio.com/items?itemName=jchannon.csharpextensions
- C# XML Documentation Comments
  https://marketplace.visualstudio.com/items?itemName=k--kato.docomment
- EditorConfig for VS Code
  https://marketplace.visualstudio.com/items?itemName=EditorConfig.EditorConfig
- vscode-icons https://marketplace.visualstudio.com/items?itemName=robertohuertasm.vscode-icons
- .NET Core Test Explorer
  https://marketplace.visualstudio.com/items?itemName=formulahendry.dotnet-test-explorer

DI/IoC & Middleware 架構

- ✓ ASP.NET Core優先考量測試，所有物件都使用內建DI初始化及存取。
- ✓ ASP.NET Core對於底層架構功能設定都靠配置和載入middleware。
- ✓ ASP.NET Core框架的middleware廣泛地使用：
  - extension method
  - fluent API notation
  - lambda expression

  做***"code as configuration"***。

DI(依賴注入，Dependency injection)

自訂類別的註冊以及物件的初始化

程式中使用的自定義物件類別，註冊至內建DI框架的服務集合(ServiceCollection)，就可以在很多地方由建構子注入(Constructor Injection)方式得到已完全初始化物件，不必自己呼叫建構子創建。

- ✓ 一定要搞清楚這三種註冊類別extension method API所初始化物件的生命週期：
  - AddScoped()
  - AddTransient()
  - AddSingleton()
- ✓ 被DI的類別如果其建構子有相依的類別，也需要在Startup.cs的 `void ConfigureServices(IServiceCollection services)` 方法內註冊。

DI(依賴注入，Dependency injection)

所有非框架本身的東西都需要在Startup.cs裡DI注入後才能用，包括自訂Filter：

https://localhost:5001/api/files/upload

An unhandled exception occurred while processing the request.

InvalidOperationException: No service for type 'AspNetCoreFileUpDown.Utils.ValidateMimeMultiPartContentFilter' has been registered.

Microsoft.Extensions.DependencyInjection.ServiceProviderServiceExtensions.GetRequiredService(IServiceProvider provider, Type serviceType)

Stack Query Cookies Headers

InvalidOperationException: No service for type 'AspNetCoreFileUpDown.Utils.ValidateMimeMultiPartContentFilter' has been registered.

Microsoft.Extensions.DependencyInjection.ServiceProviderServiceExtensions.GetRequiredService(IServiceProvider provider, Type serviceType)
Microsoft.AspNetCore.Mvc.ServiceFilterAttribute.CreateInstance(IServiceProvider serviceProvider)
Microsoft.AspNetCore.Mvc.Internal.DefaultFilterProvider.ProvideFilter(FilterProviderContext context, FilterItem filterItem)
Microsoft.AspNetCore.Mvc.Internal.DefaultFilterProvider.OnProvidersExecuting(FilterProviderContext context)
Microsoft.AspNetCore.Mvc.Internal.FilterFactory.CreateUncachedFiltersCore(IFilterProvider[] filterProviders, ActionContext actionContext, List<FilterItem> filterItems)

DI(依賴注入，Dependency injection)

自訂類別的註冊以及物件的初始化

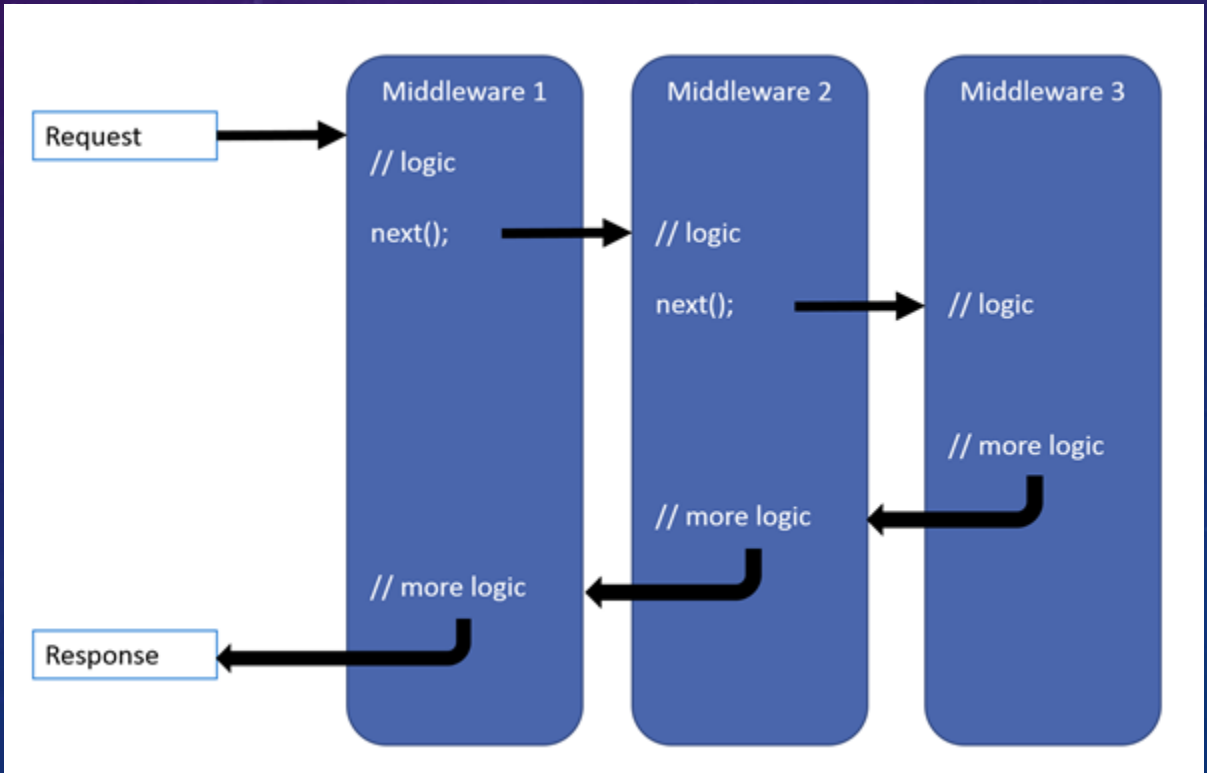
- ✓ 預設能使用注入的自訂DI類別物件起始點是從Startup.cs的 `void Configure(IApplicationBuilder app, IHostingEnvironment env)` 開始，被注入的附加DI參數列在原本的app, env參數之後。

```
// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)
{
    loggerFactory.AddConsole(Configuration.GetSection("Logging"));
    loggerFactory.AddDebug();

    if (env.IsDevelopment())
```

- 不使用附加DI參數/建構子注入，而仍能取得DI物件的範例：https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection#resolve-a-scoped-service-within-the-application-scope
- ASP.NET Core Dependency Injection Deep Dive: https://joonasw.net/view/aspnet-core-di-deep-dive

Middleware架構

- Middleware(中間件、中介軟體)的概念是仿效Node.js的express框架所講的middleware概念。
- 類似ASP.NET Web API的HTTP Message Handler架構，用來控制/改變/增加ASP.NET Core處理HTTP request/response的流程。



Middleware架構

- 幾乎所有對Server增加額外功能的方式都是靠設置&添加middleware。
- 如果不套用MVC框架，全靠把處理邏輯寫在middleware也可以。
- 官方提供的middleware列表：https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?tabs=aspnetcore2x#built-in-middleware
- 自行撰寫Middleware要注意：
  1. 在middleware的流程邏輯內只能呼叫一次**`next.Invoke(context)`**。
  2. middleware修改header/body內容也得符合HTTP規範，否則asp.net core底層會報錯。
  3. 在async方法內要呼叫**`await next.Invoke(context)`**，而不是**await next(context)**。
- 自製HTTP request/response 紀錄器的範例：http://bit.ly/2LiHWy6

設定檔與 Options Pattern

- ✓ ASP.NET Core的設定檔改為JSON格式，沒有先前的Web.config transform。
- ✓ 設定資料可由ASP.NET Core由環境變數/設定檔/EF載入，對應到由DI注入的強型別物件上。

- 設定檔對應每個組態(Debug/Release)是個別獨立的，沒有XML transform tool在執行時繼承別的設定檔內容轉換產生(https://msdn.microsoft.com/en-us/library/dd465326.aspx)的機制，但在Startup.cs程式碼內載入時可以自行調整載入順序或載入檔名的邏輯(AddJsonFile() & AddXmlFile() API)。

```
public static IConfigurationBuilder AddConfig(WebHostBuilderContext context, IConfigurationBuilder configurationBuilder, string configName)
{
    var env = context.HostingEnvironment;

    configurationBuilder
        .AddJsonFile($"{configName}.json", optional: true, reloadOnChange: true)
        .AddJsonFile($"{configName}.{env.EnvironmentName}.json", optional: true, reloadOnChange: true);

    return configurationBuilder;
}
```

設定檔與 Options Pattern

- ASP.NET Core提供Secret Manager全域指令工具於開發時期存於各台電腦上不同的設定來動態轉換設定值，適合管理DB連線密碼/API Key等敏感資訊：https://docs.microsoft.com/en-us/aspnet/core/security/app-secrets
- 從設定檔/系統變數/讀進系統的設定值，對應抄寫到強型別DI注入物件的Value屬性上，稱為Options Pattern： https://docs.microsoft.com/en-us/aspnet/core/fundamentals/configuration/options
  - 分隔不同的設定值群組。
  - 確保當設定檔沒有寫時，在程式碼內還有類別宣告的預設值。

設定檔與 Options Pattern

1. 宣告強型別類別且可在建構子設定屬性預設值

FileUploadConfig.cs

```
namespace AspNetCoreFileUpDown.Configs
{
    public class FileUploadConfig
    {
        public FileUploadConfig()
        {
            ServerUploadFolder = "upload";
            UploadNamingPattern = "{file_name}";
            OverwriteExisted = true;
        }

        public string ServerUploadFolder { get; set; }

        public string UploadNamingPattern { get; set; }

        public bool OverwriteExisted { get; set; }
    }
}
```

設定檔與 Options Pattern

2. Startup.cs的
   public void ConfigureServices(IServiceCollection services)
   內使用Configure<TOptions>() 擴充方法API將該類別註冊到內建DI：

```
public void ConfigureServices(IServiceCollection services)
{
    //Register Config Options Class
    services.Configure<FileUploadConfig>(Configuration.GetSection(nameof(FileUploadConfig)));
```

3. 使用建構子注入的方式取得已從設定檔載入設定值的該類別實體：

```
public class FileUploadController : Controller
{
    private readonly IFileRepository _fileRepo;

    private readonly FileUploadConfig _fileUploadConfig;

    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public FileUploadController(IFileRepository fileRepository, IOptions<FileUploadConfig> options)
    {
        _fileRepo = fileRepository;

        _fileUploadConfig = options.Value;
    }
```

設定檔與 Options Pattern

- ✓ 在Startup.cs註冊各類DI物件服務的
  `public void ConfigureServices(IServiceCollection services){}`
  這個生命週期階段還無法使用Options Pattern類別，此處如果要讀設定檔，只能先用key-value的方式從建構子注入的IConfiguration讀出設定字串值。

```
public void ConfigureServices(IServiceCollection services)
{
    //option configuraiton
    SetupSettings(services, Configuration);

    //Database service configuraiton
    ConfigDbService(services, Configuration);
```

webapi

```
1 reference
private static void ConfigDbService(IServiceCollection services, IConfiguration configuration)
{
    var dbType = configuration["DbConfig:DB_Type"];
    var dbConn = configuration["DbConfig:DB_Conn"];

    if (dbType == "in-memory")
    {
        services.AddDbContext<[redacted]DbContext>(options =>
            options.UseInMemoryDatabase(dbConn)
        );
    }
```

EFcore Code-First / DB provider for No-SQL

- ✓ EFCore給ASP.NET Core的開發方式目前只有Code-First 的指令列操作
- ✓ Code-First的CLI指令可產生更新DB用的SQL Script。
- ✓ In-Memory DB可用來做整合測試。
- ✓ 未來會有官方支援的No-SQL DB Provider，讓底層不再只有SQL based DB。

- EF Core(Entity Framework Core)的全域指令在.NET Core 2.1之後放進Global Tool，不必再額外安裝 `<DotNetCliToolReference>` 的Nuget套件：https://docs.microsoft.com/en-us/ef/core/what-is-new/ef-core-2.1#new-dotnet-ef-global-tool
- DB driver (Database Provider)直接以Nuget套件方式安裝，避免需要各種安裝driver於執行OS系統上的權限和衝突。

EFCore Code-First

- 務必更新Runtime Nuget套件，不然現在 .NET Core 2.0專案會有這個問題：https://stackoverflow.com/a/48598679

```
D:\repo\AspNetCoreFileUploadFileTable\src\DataAccess>dotnet ef migrations add my1st
T
he specified framework version '2.0' could not be parsed
The specified framework 'Microsoft.NETCore.App', version '2.0' was not found.
  - Check application dependencies and target a framework version installed at:
      C:\Program Files\dotnet\
  - Installing .NET Core prerequisites might help resolve this problem:
      http://go.microsoft.com/fwlink/?LinkID=798306&clcid=0x409
  - The .NET Core framework and SDK can be installed from:
      https://aka.ms/dotnet-download
  - The following versions are installed:
      1.0.1 at [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
      1.0.5 at [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
      1.1.2 at [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
      2.0.0 at [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
      2.0.3 at [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
      2.0.5 at [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
      2.0.6 at [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
      2.0.7 at [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
      2.1.0-preview2-26406-04 at [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
```

EFcore Code-First

- EF Core指令：http://docs.microsoft.com/en-us/ef/core/miscellaneous/cli/dotnet#commands

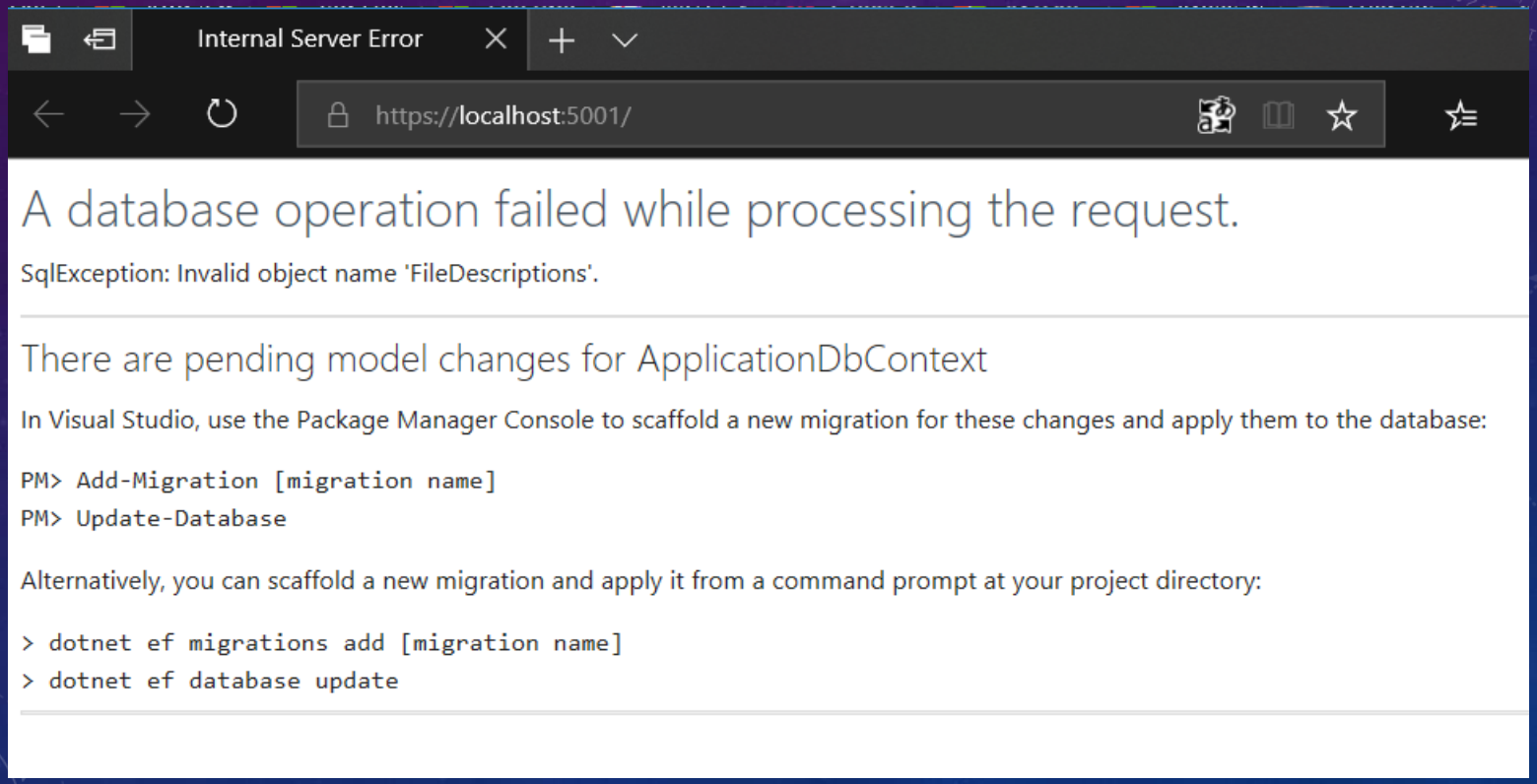
```
dotnet ef dbcontext list
dotnet ef dbcontext info
dotnet ef migrations add [tag_name] -c [DbContextClassName]
dotnet ef migrations list -c [DbContextClassName]
dotnet ef migrations remove -c [DbContextClassName]
dotnet ef database update [tag_name]或0
dotnet ef migrations script [from_tag_name] [to_tag_name] -o [generated_sql_file]
```

✓ 這些指令在ASP.NET Core 2.0之後會去爬Program.cs裡原本ASP.NET Core專案範本的*預設建立**WebHost**方法*，如果沒有的話，指令列會抓不到DBContext類別而無法運作。

- **ASP.NET Core 2.0**：
  `public static IWebHost BuildWebHost(string[] args)`
- **ASP.NET Core 2.1**：
  `public static IWebHostBuilder CreateWebHostbuilder(string[] args)`

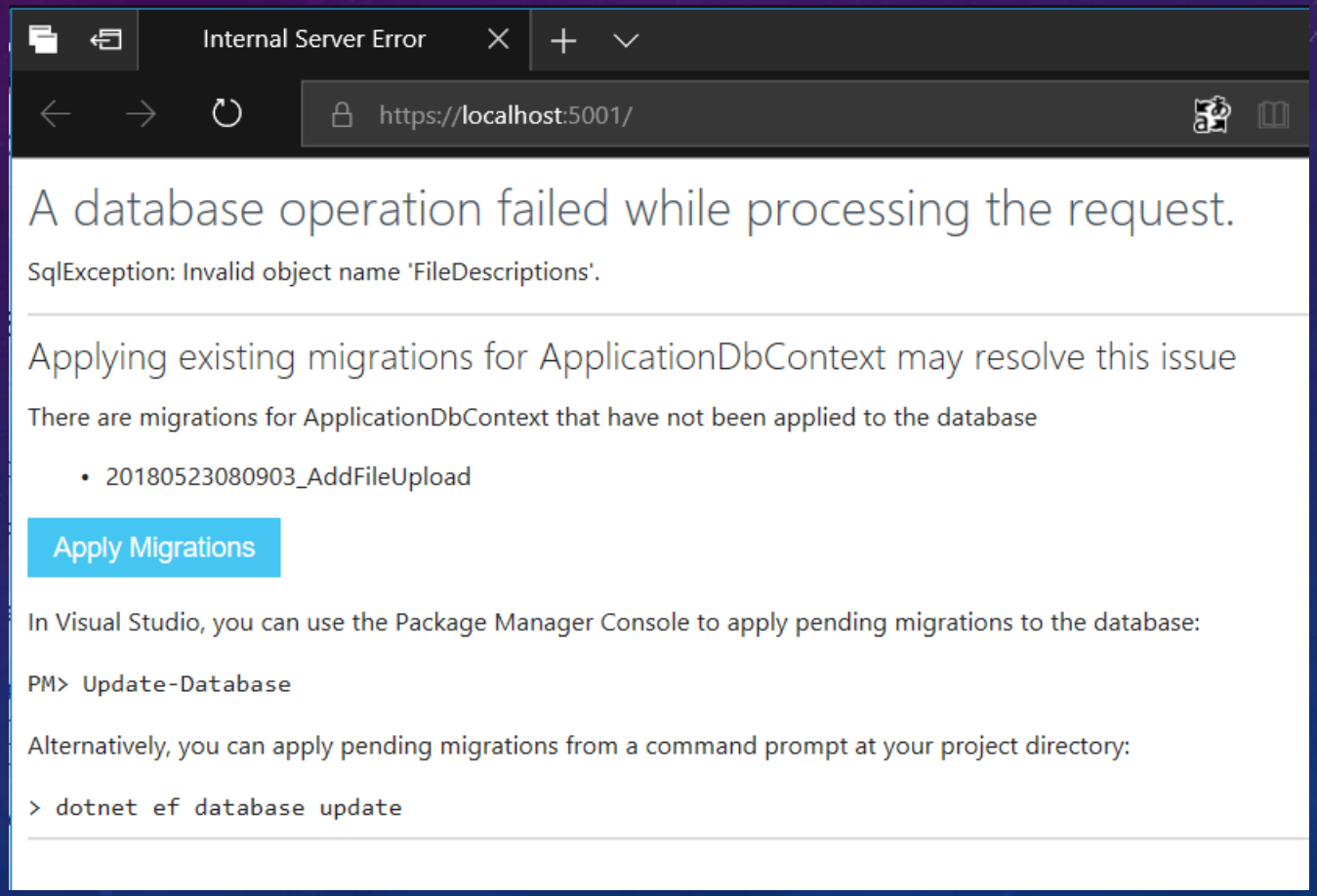
EFCore Code-First

- 如果沒有用全域指令建立migrations，在開發階段執行時會有如下錯誤：



EFCore Code-First

- 如果是沒有套用migrations，開發階段執行時可以直接在網頁上套用：



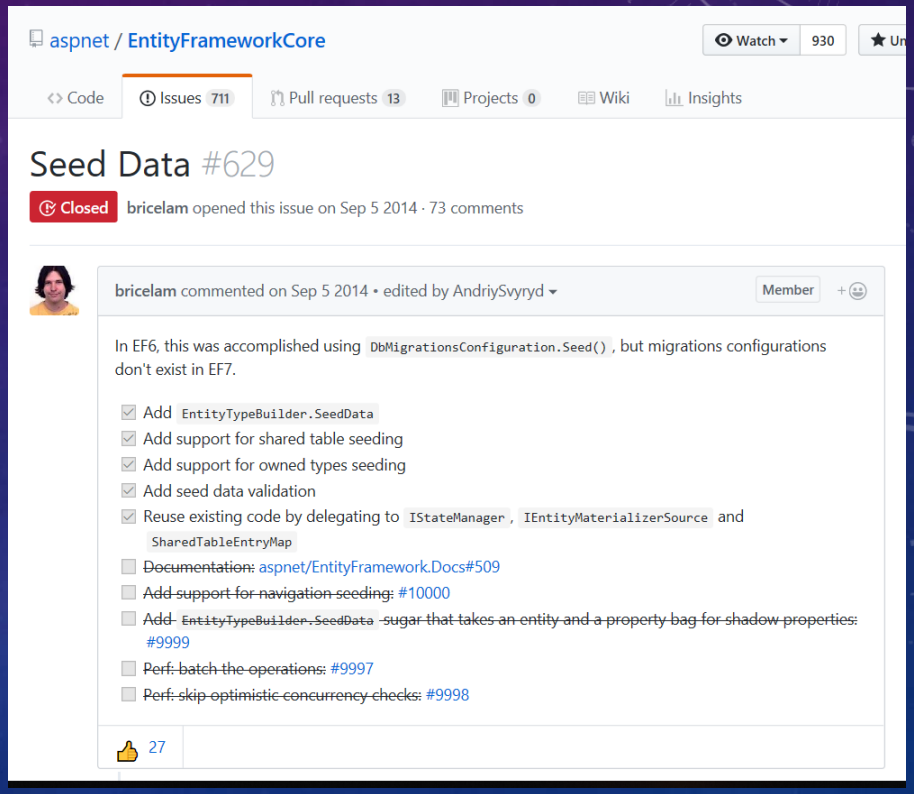
EFCore Code-First

- 效果同等於下dotnet ef database update指令：

```
PS D:\repo\AspNetCoreFileUpDown\AspNetCoreFileUpDown> dotnet ef database update
info: Microsoft.EntityFrameworkCore.Infrastructure[10403]
      Entity Framework Core 2.1.0-rc1-32029 initialized 'ApplicationDbContext' using provider
'Microsoft.EntityFrameworkCore.SqlServer' with options: None
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (8ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT OBJECT_ID(N'[__EFMigrationsHistory]');
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (1ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT OBJECT_ID(N'[__EFMigrationsHistory]');
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (2ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT [MigrationId], [ProductVersion]
      FROM [__EFMigrationsHistory]
      ORDER BY [MigrationId];
info: Microsoft.EntityFrameworkCore.Migrations[20402]
      Applying migration '20180523080903_AddFileUpload'.
Applying migration '20180523080903_AddFileUpload'.
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (6ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      CREATE TABLE [FileDescriptions] (
          [Id] int NOT NULL IDENTITY,
          [FileName] nvarchar(max) NULL,
          [Description] nvarchar(max) NULL,
          [CreatedTimestamp] datetime2 NOT NULL,
          [UpdatedTimestamp] datetime2 NOT NULL,
          [ContentType] nvarchar(max) NULL,
          CONSTRAINT [PK_FileDescriptions] PRIMARY KEY ([Id])
      );
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (3ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      INSERT INTO [__EFMigrationsHistory] ([MigrationId], [ProductVersion])
      VALUES (N'20180523080903_AddFileUpload', N'2.1.0-rc1-32029');
Done.
PS D:\repo\AspNetCoreFileUpDown\AspNetCoreFileUpDown>
```

EFCore Code-First

- ✓ 新的EF Core 2.1 提供Seeding Data API在關聯資料上會有問題，建議還是用自己寫code的方式塞初始化資料：https://github.com/aspnet/EntityFrameworkCore/issues/629



EFCore Code-First / DB provider for No-SQL

- DB provider讓底層的儲存資料庫系統可以抽換：
  https://docs.microsoft.com/en-us/ef/core/providers/

```
services.AddDbContext<FileContext>(options =>
    //options.UseSqlServer(
    //    sqlConnectionString,
    //    b => b.MigrationsAssembly("AspNetCoreFileUploadFileTable")
    //)
    options.UseSqlite(sqlConnectionString, b=>b.MigrationsAssembly("AspNetCoreFileUploadFileTable"))
);
```

- 用In-Memory DB的配置做整合測試： https://docs.microsoft.com/en-us/ef/core/miscellaneous/testing/in-memory
  → In-Memory DB不是關聯式資料庫

EFcore Code-First / DB provider for No-SQL

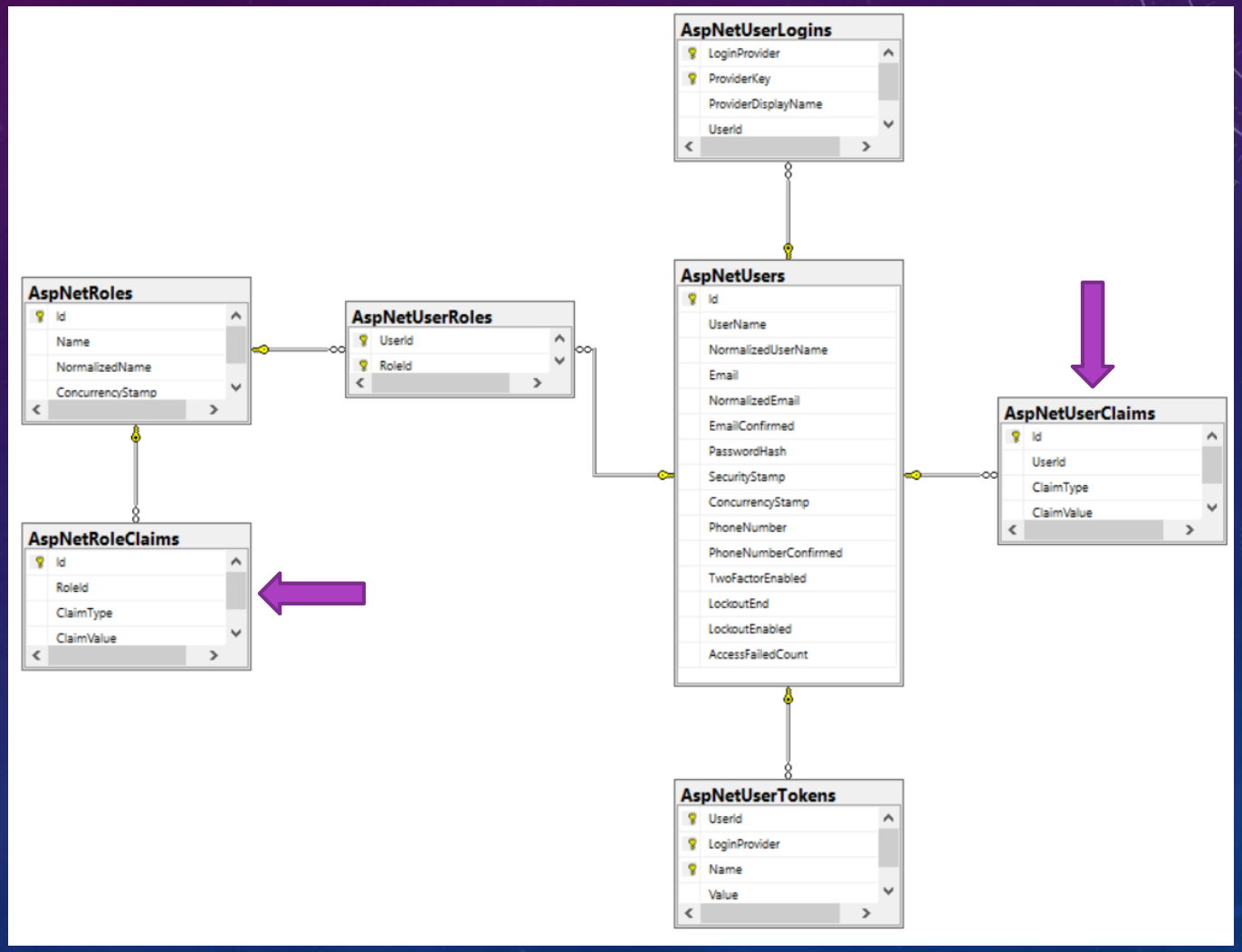
- Azrue CosmosDB (前身叫DocumentDB) 的DB Provider 之後官方會釋出：https://docs.microsoft.com/en-us/ef/core/providers/#future-providers
- 目前有第三方的MongoDB Provider可以使用：https://github.com/crhairr/EntityFrameworkCore.MongoDb

ASP.NET Core Identity

- ✓ **Claim-base authentication**讓單一帳號可使用多種不同驗證方式登入。
- ✓ ASP.NET Core 2.1提供的 Microsoft.AspNetCore.Identity.UI 套件可產生符合GDPR法規要求的使用者資料下載和刪除功能帳號管理頁面。
- ✓ ASP.NET Core Identity底層可使用非SQL Server的資料庫系統儲存。

- ✓ **Claim-base authentication** (宣告式身分識別)
  - 建議一定要看這篇介紹何謂Claim-base auth: https://andrewlock.net/introduction-to-authentication-with-asp-net-core/
  - 單個ASP.NET Core Identity User帳號、Role角色權限可以綁多個Claim。
  - 藉由確認多個Claim Type & Claim Value都符合的方式來驗證是該使用者。

ASP.NET Core Identity



ASP.NET Core Identity

- 預設User, Role使用PK欄位型態是string (雖然內部存的是GUID字串)，可由繼承 `IdentityUser<T>`, `IdentityRole<T>` 的方式改用別種資料型態存。

✓ 但如果已經將原本的EF Core套用到DB上，得先回復DB到完全沒有一開始預設ASP.NET Core專案範本所做的db migration
(也就是得下指令：`dotnet ef database update 0`)，
且將一開始專案範本建的EF Core DB migration移除，才有辦法成功更新資料庫。

```
      ALTER TABLE [AspNetUserTokens] ALTER COLUMN [LoginProvider] nvarchar(450) NOT NULL;
fail: Microsoft.EntityFrameworkCore.Database.Command[20102]
      Failed executing DbCommand (19ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      DECLARE @var2 sysname;
      SELECT @var2 = [d].[name]
      FROM [sys].[default_constraints] [d]
      INNER JOIN [sys].[columns] [c] ON [d].[parent_column_id] = [c].[column_id] AND [d].[parent_object_id] = [c].[object_id]
      WHERE ([d].[parent_object_id] = OBJECT_ID(N'[AspNetUserTokens]') AND [c].[name] = N'UserId');
      IF @var2 IS NOT NULL EXEC(N'ALTER TABLE [AspNetUserTokens] DROP CONSTRAINT [' + @var2 + '];');
      ALTER TABLE [AspNetUserTokens] ALTER COLUMN [UserId] uniqueidentifier NOT NULL;
System.Data.SqlClient.SqlException (0x80131904): The object 'PK_AspNetUserTokens' is dependent on column 'UserId'.
The object 'FK_AspNetUserTokens_AspNetUsers_UserId' is dependent on column 'UserId'.
ALTER TABLE ALTER COLUMN UserId failed because one or more objects access this column.
   at System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)
   at System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)
   at System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean
asyncClose)
   at System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet
bulkCopyHandler, TdsParserStateObject stateObj, Boolean& dataReady)
   at System.Data.SqlClient.SqlCommand.RunExecuteNonQueryTds(String methodName, Boolean async, Int32 timeout, Boolean asyncWrite)
   at System.Data.SqlClient.SqlCommand.InternalExecuteNonQuery(TaskCompletionSource`1 completion, Boolean sendToPipe, Int32 timeout, Boolean
asyncWrite, String methodName)
```

Account.cs

AspNetCoreFileUpDown | AspNetCoreFileUpDown.Models.Account

```
5   {
6       public class Account : IdentityUser<Guid>
7       {
8       }
9   }
10
```

100 %

AccountGroup.cs

AspNetCoreFileUpDown | AspNetCoreFileUpDown.Models.AccountGroup

```
5   {
6       public class AccountGroup : IdentityRole<Guid>
7       {
8       }
9   }
```

100 %

ApplicationDbContext.cs

AspNetCoreFileUpDown | AspNetCoreFileUpDown.Data.ApplicationDbContext

```
9   {
10      public class ApplicationDbContext : IdentityDbContext<Account, AccountGroup, Guid>
11      {
12          public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
13              : base(options)
14          {
15          }
16
17          public DbSet<FileDescription> FileDescriptions { get; set; }
18      }
19  }
20
```

- 記得改View上原本專案範本所注入的SignInManager和UserManager型態，

_LoginPartial.cshtml

```
1  @using Microsoft.AspNetCore.Identity
2
3  @inject SignInManager<Account> SignInManager
4  @inject UserManager<Account> UserManager
5
6  @if (SignInManager.IsSignedIn(User))
7  {
8      <form asp-area="Identity" asp-page="/Account/Logout"
9          <ul class="nav navbar-nav navbar-right">
10             <li>
```

- 否則執行時會爆出一個乍看之下很奇怪的錯誤：

An unhandled exception occurred while processing the request.

InvalidOperationException: No service for type 'Microsoft.AspNetCore.Identity.UserManager`1[Microsoft.AspNetCore.Identity.IdentityUser]' has been registered.

Microsoft.Extensions.DependencyInjection.ServiceProviderServiceExtensions.GetRequiredService(IServiceProvider provider, Type serviceType)

Stack Query Cookies Headers

InvalidOperationException: No service for type 'Microsoft.AspNetCore.Identity.UserManager`1[Microsoft.AspNetCore.Identity.IdentityUser]' has been registered.

```
Microsoft.Extensions.DependencyInjection.ServiceProviderServiceExtensions.GetRequiredService(IServiceProvider provider, Type serviceType)
Microsoft.AspNetCore.Mvc.Razor.Internal.RazorPagePropertyActivator+<>c__DisplayClass8_0.<CreateActivateInfo>b__1(ViewContext context)
Microsoft.Extensions.Internal.PropertyActivator<TContext>.Activate(object instance, TContext context)
Microsoft.AspNetCore.Mvc.Razor.Internal.RazorPagePropertyActivator.Activate(object page, ViewContext context)
Microsoft.AspNetCore.Mvc.Razor.RazorPageActivator.Activate(IRazorPage page, ViewContext context)
Microsoft.AspNetCore.Mvc.Razor.RazorView.RenderPageCoreAsync(IRazorPage page, ViewContext context)
Microsoft.AspNetCore.Mvc.Razor.RazorView.RenderPageAsync(IRazorPage page, ViewContext context, bool invokeViewStarts)
Microsoft.AspNetCore.Mvc.Razor.RazorView.RenderAsync(ViewContext context)
Microsoft.AspNetCore.Mvc.TagHelpers.PartialTagHelper.RenderPartialViewAsync(TextWriter writer, object model)
Microsoft.AspNetCore.Mvc.TagHelpers.PartialTagHelper.ProcessAsync(TagHelperContext context, TagHelperOutput output)
Microsoft.AspNetCore.Razor.Runtime.TagHelpers.TagHelperRunner.RunAsync(TagHelperExecutionContext executionContext)
AspNetCore.Views_Shared__Layout.<ExecuteAsync>b__46_1()
Microsoft.AspNetCore.Razor.Runtime.TagHelpers.TagHelperExecutionContext.SetOutputContentAsync()
AspNetCore.Views_Shared__Layout.ExecuteAsync()
Microsoft.AspNetCore.Mvc.Razor.RazorView.RenderPageCoreAsync(IRazorPage page, ViewContext context)
```

ASP.NET Core Identity – Identity UI

- Identity UI : Razor UI包裝在一個使用預設meta package就安裝的Nuget套件：Microsoft.AspNetCore.Identity.UI
  https://www.nuget.org/packages/Microsoft.AspNetCore.Identity.UI
- 提供最基本滿足歐盟**GDPR(General Data Protection Regulation)**法規規範的使用者帳號資料管理功能頁面。
  https://www.ithome.com.tw/news/116876

- AspNetCoreFileUpDown

https://localhost:5001/Identity/Account/Manage/PersonalData

AspNetCoreFileUpDown Home About Contact Hello abc@123.def! Logout

Manage your account

Change your account settings

Profile

Password

Two-factor authentication

Personal data

Personal Data

Your account contains personal data that you have given us. This page allows you to download or delete that data.

Deleting this data will permanently remove your account, and this cannot be recovered.

Download

Delete

© 2018 - AspNetCoreFileUpDown

https://localhost:5001/Identity/Account/Manage/DownloadPersonalData

ASP.NET Core Identity – Identity UI

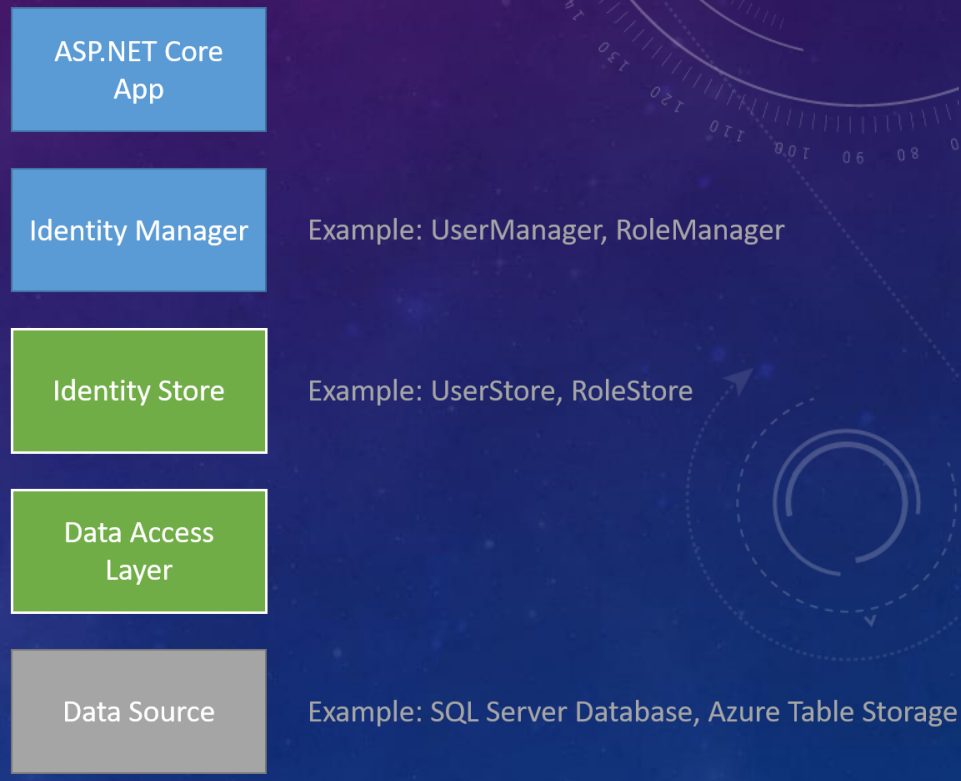
- Identity UI啟用方法：

```
37            services.AddDbContext<ApplicationDbContext>(options =>
38                options.UseSqlServer(
39                    Configuration.GetConnectionString("DefaultConnection")));
40
41            services.AddDefaultIdentity<IdentityUser>()
42                .AddEntityFrameworkStores<ApplicationDbContext>()
43                .AddDefaultUI()
44                .AddDefaultTokenProviders();
45
46            services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
47        }
```

- Identity UI除了單用Nuget套件方式之外，也可產Code方式以便客製化：https://blogs.msdn.microsoft.com/webdev/2018/03/02/aspnetcore-2-1-identity-ui/

ASP.NET Core Identity - Custom storage providers

- ASP.NET Core Identity預設底層是用EF Core + SQL Server，實際設計有抽換底層的功能：https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity-custom-storage-providers
- 第三方的儲存於Azure DocumetDB storage provider: https://github.com/tracker086/DocumentDB.AspNet.Identity



Unit/Integration Test Utility

- ✓ ASP.NET Core的設計原則是測試優先，不能測的甚至是API都會砍掉。
- ✓ EF Core In-Memory DB不用做 add migration/db update就能整合測試。
- ✓ Microsoft.AspNetCore.TestHost套件裡的TestServer提供測試用`HttpClient`, `WebSocketClient`，不用真正binding系統TCP port就能跑整合測試。
- ✓ xUnit的測試案例在不同Test Class是平行同步跑的，所以寫測試時要注意。

- IFormFile的`SaveAsAsync()`因為沒辦法寫測試，所以被砍掉了：https://github.com/aspnet/HttpAbstractions/issues/610
- 對MVC Controller Action方法做單元測試可參考官方說明文件：https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/testing

Unit/Integration Test Utility

- EF Core In-Memory DB(安裝Microsoft.EntityFrameworkCore.InMemory套件)
  可以在開發時期就用它快速修正DB Schema，跑整合測試。
- 但In-Memory DB嚴格來說不是Relational資料庫，有些EF Core指令不能用：

```
PS D:\repo\granden\server\webapi\webapi> dotnet ef dbcontext info
System.InvalidOperationException: Relational-specific methods can only be used when the context is using a relational database provider.
   at Microsoft.EntityFrameworkCore.RelationalDatabaseFacadeExtensions.GetRelationalService[TService](IInfrastructure`1 databaseFacade)
   at Microsoft.EntityFrameworkCore.RelationalDatabaseFacadeExtensions.GetDbConnection(DatabaseFacade databaseFacade)
   at Microsoft.EntityFrameworkCore.Design.Internal.DbContextOperations.GetContextInfo(String contextType)
   at Microsoft.EntityFrameworkCore.Design.OperationExecutor.GetContextInfoImpl(String contextType)
   at Microsoft.EntityFrameworkCore.Design.OperationExecutor.GetContextInfo.<>c__DisplayClass0_1.<.ctor>b__0()
   at Microsoft.EntityFrameworkCore.Design.OperationExecutor.OperationBase.<>c__DisplayClass3_0`1.<Execute>b__0()
   at Microsoft.EntityFrameworkCore.Design.OperationExecutor.OperationBase.Execute(Action action)
Relational-specific methods can only be used when the context is using a relational database provider.
```

Unit/Integration Test Utility

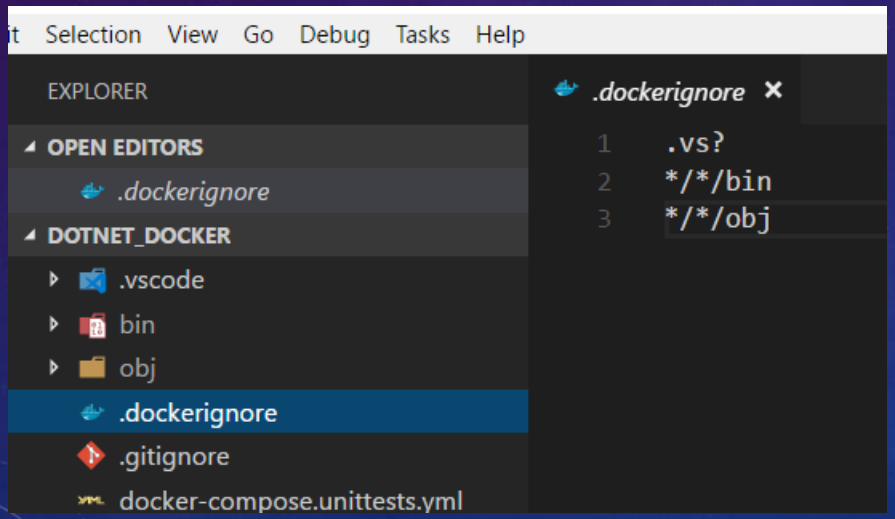
- Microsoft.AspNetCore.TestHost套件是專門用來做整合測試用的：
  - TestServer.CreateClient() 產生連TestHost的HttpClient
  - TestServer.CreateWebSocketClient() 產生連TestHost的WebSocketClient

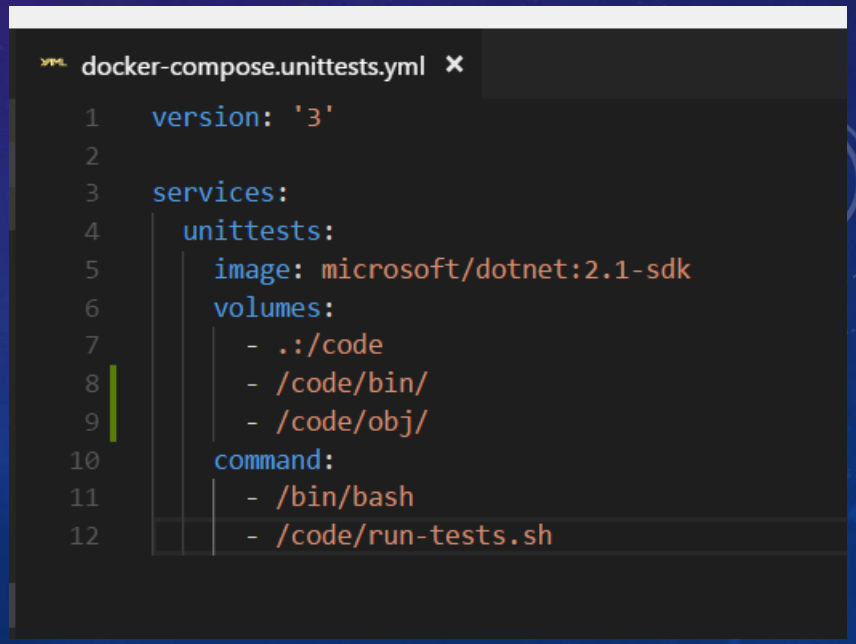
  TestHost的TestServer跑起來的時候不會真正占用到系統的TCP port。

  https://blogs.msdn.microsoft.com/webdev/2017/12/07/testing-asp-net-core-mvc-web-apps-in-memory/
- 可配合Test框架的Fixture來做，例如用xUnit的例子：https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/testing#accessing-views
- xUnit寫整合測試時要注意，不同Test Class裡的測試案例會平行同步跑：https://xunit.github.io/docs/running-tests-in-parallel.html

Unit/Integration Test Utility

- 使用docker來跑測試時，需要加 .dockerignore檔(https://docs.docker.com/engine/reference/builder/#dockerignore-file) 和在docker-compose.yml跑測試的docker服務的volume上做一些設定(https://docs.docker.com/compose/compose-file/#short-syntax-3)，以避免專案內的bin & obj目錄在不同系統上會互相干擾。
  範例： https://github.com/windperson/demo_docker_dotnetcore_test





使用docker跑整合測試出現的現象，原因是bin & obj目錄在不同OS(Win/Linux)會產生不同的檔案造成C#擴充套件誤判。

dotnet_docker - Visual Studio Code

File Edit Selection View Go Debug Tasks Help

EXPLORER

OPEN EDITORS

DOTNET_DOCKER

- .vscode
  - launch.json
  - settings.json
  - tasks.json
- bin
- obj
- .dockerignore U
- .gitignore
- docker-compose.unittests.yml M
- dotnet_docker.csproj M
- Example.cs +9
- run-tests.sh M
- UnitTest1.cs 8, M

.NET TEST EXPLORER

- dotnet_docker

PROBLEMS 40 OUTPUT DEBUG CONSOLE TERMINAL 1: powershell

```
  You are working with a preview version of the .NET Core SDK. You can define the SDK version via a global.jso
n file in the current project. More at https://go.microsoft.com/fwlink/?linkid=869452
  dotnet_docker -> /code/bin/Debug/netcoreapp2.1/dotnet_docker.dll
Build completed.

Test run for /code/bin/Debug/netcoreapp2.1/dotnet_docker.dll(.NETCoreApp,Version=v2.1)
Microsoft (R) Test Execution Command Line Tool Version 15.7.0-preview-20180221-13
Copyright (c) Microsoft Corporation.  All rights reserved.

Starting test execution, please wait...

Total tests: 2. Passed: 2. Failed: 0. Skipped: 0.
Test Run Successful.
Test execution time: 2.4032 Seconds
PS D:\myTemp\dotnet_docker>
```

There are unresolved dependencies from 'dotnet_docker.csproj'.
Please execute the restore command to continue.

Source: C# (Extension) Restore

master* 40 0 .NET Core Launch (console) (dotnet_docker) Watch dotnet_docker 1

Deployment & 容器化(dockerize)

- ✓ Framework-dependent deployments (FDD)佈署需在環境上安裝對應版本的runtime。
- ✓ Self-contained deployments (SCD)佈署雖然不需要安裝runtime，但要注意Visual Studio佈署精靈的bug，目前Windows上產Linux的SCD會有問題。
- ✓ IIS佈署可藉由改產生的web.config之中的設定值來除錯。

- FDD佈署方式需要在佈署環境安裝runtime。
- SCD佈署方式將整個產出目錄copy到佈署環境就可執行。
- ✓ `dotnet run`指令執行時用的Properties\LaunchSettings.json設定檔是開發時期用的，不是拿來佈署正式環境的。

Deployment - SCD佈署

- ✓ 在Windows上的Visual Studio 2017的.NET Core 2.1 rc1 做SCD佈署到Linux平台仍然會有系統assembly載入錯誤的問題：
  https://github.com/dotnet/sdk/issues/1502

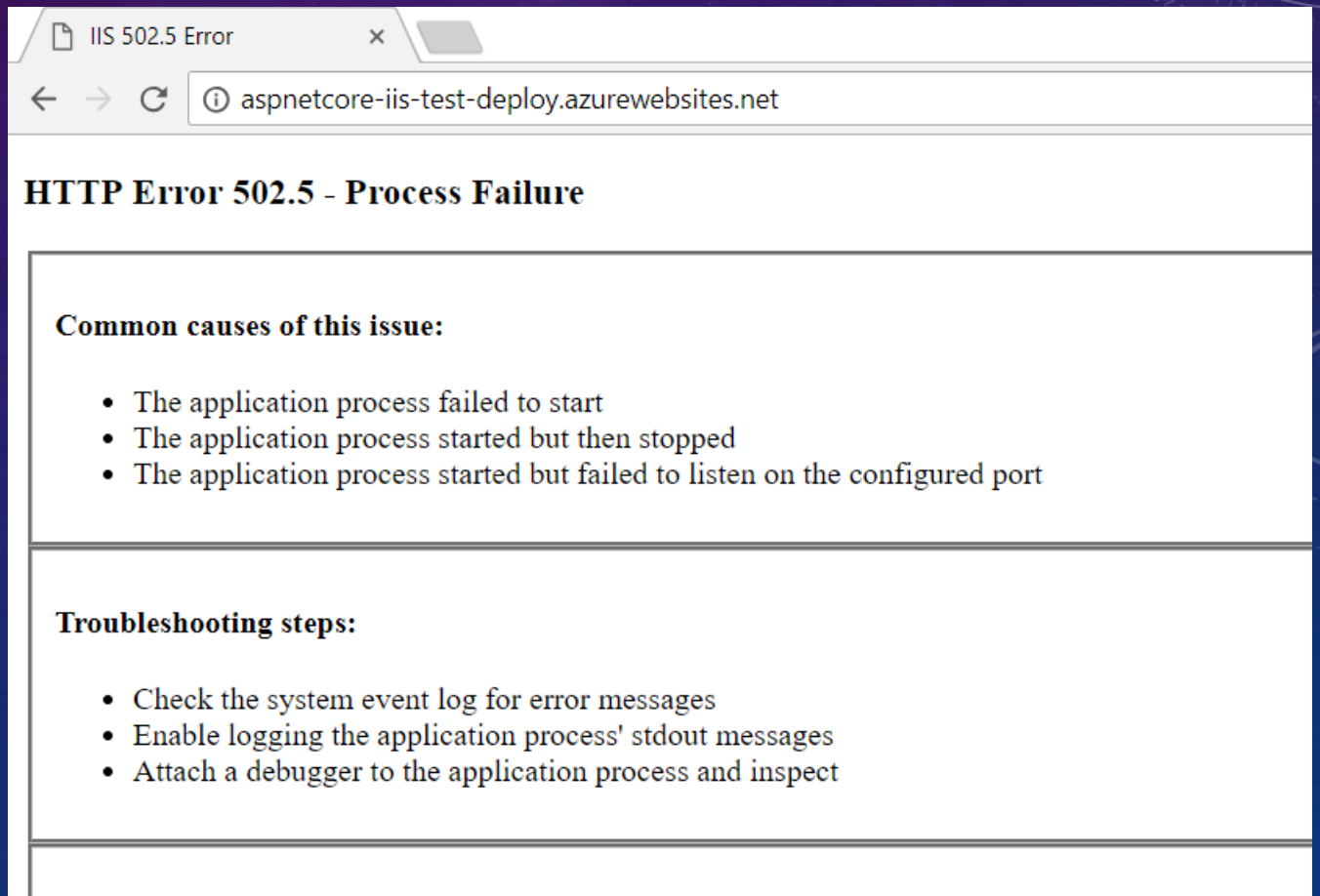
```
vmusr@vm-ubuntu1804:/media/sf_repo/granden/server/webapi/publish$ ./webapi

Unhandled Exception: System.IO.FileLoadException: Could not load file or assembly 'System.Threading, Version=4.1.1.0, Culture=neutral
, PublicKeyToken=b03f5f7f11d50a3a'. The located assembly's manifest definition does not match the assembly reference. (Exception from
 HRESULT: 0x80131040)
   at System.Console.set_OutputEncoding(Encoding value)
   at webapi.Program.Main(String[] args)
Aborted (core dumped)
vmusr@vm-ubuntu1804:/media/sf_repo/granden/server/webapi/publish$
```

- 建議還是使用打指令的方式產生：
  `dotnet publish -f netcoreapp2.1 -r linux-x64 --self-contained -o .\publish`

- ✓ 佈署環境的系統要求(含需安裝軟體)也得留意：
  http://github.com/dotnet/core/blob/master/Documentation/prereqs.md

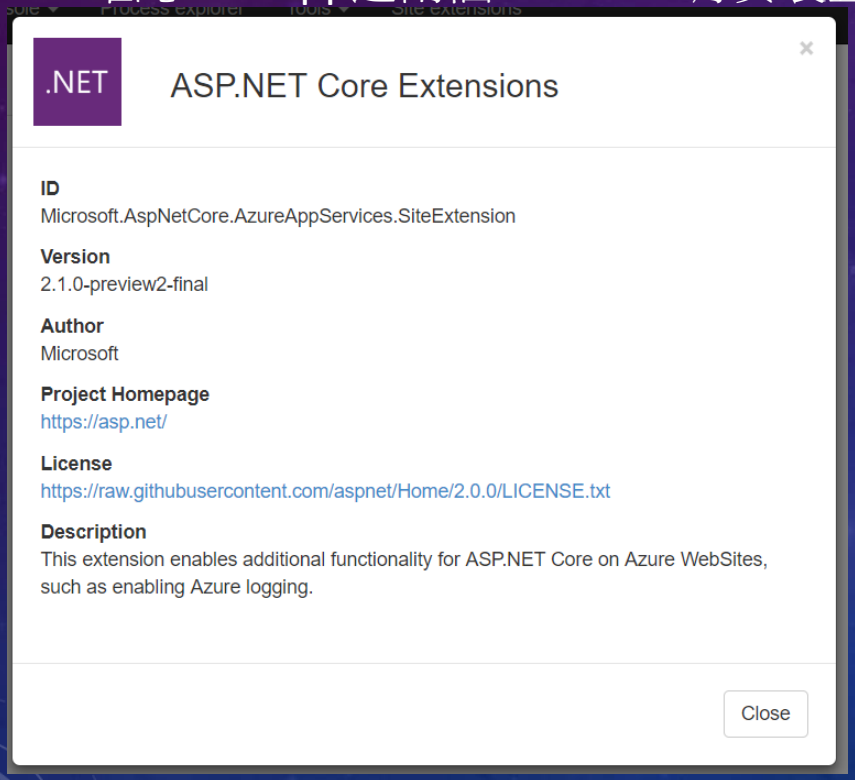
Deployment – Azure WebApp佈署troubleshoot

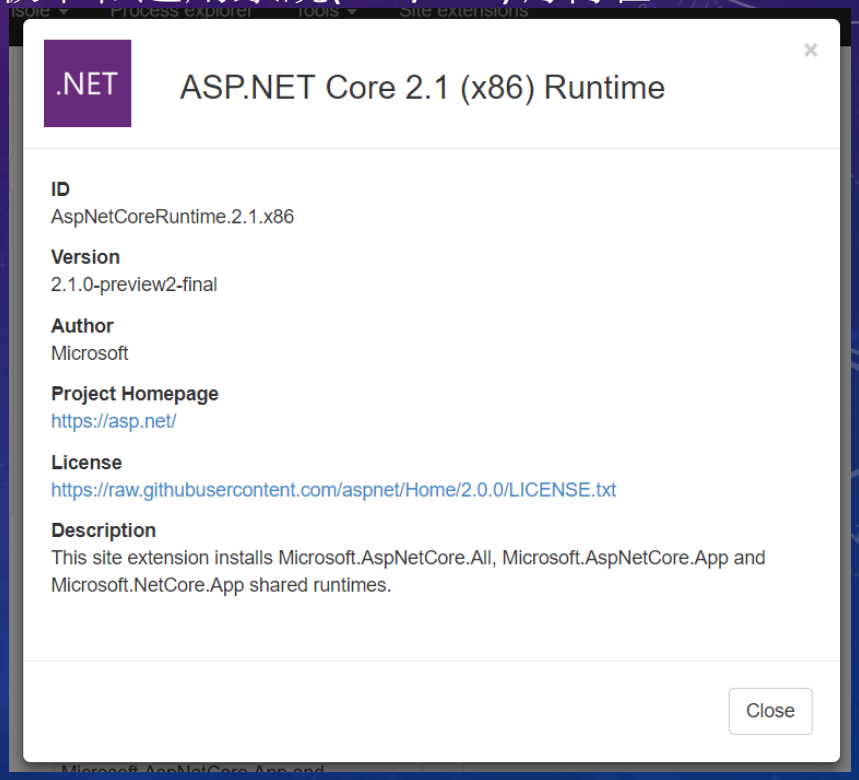
- 佈署至Azure WebApp(Windows)碰到這個IIS 502.5 Error：



Deployment – Azure WebApp佈署troubleshoot

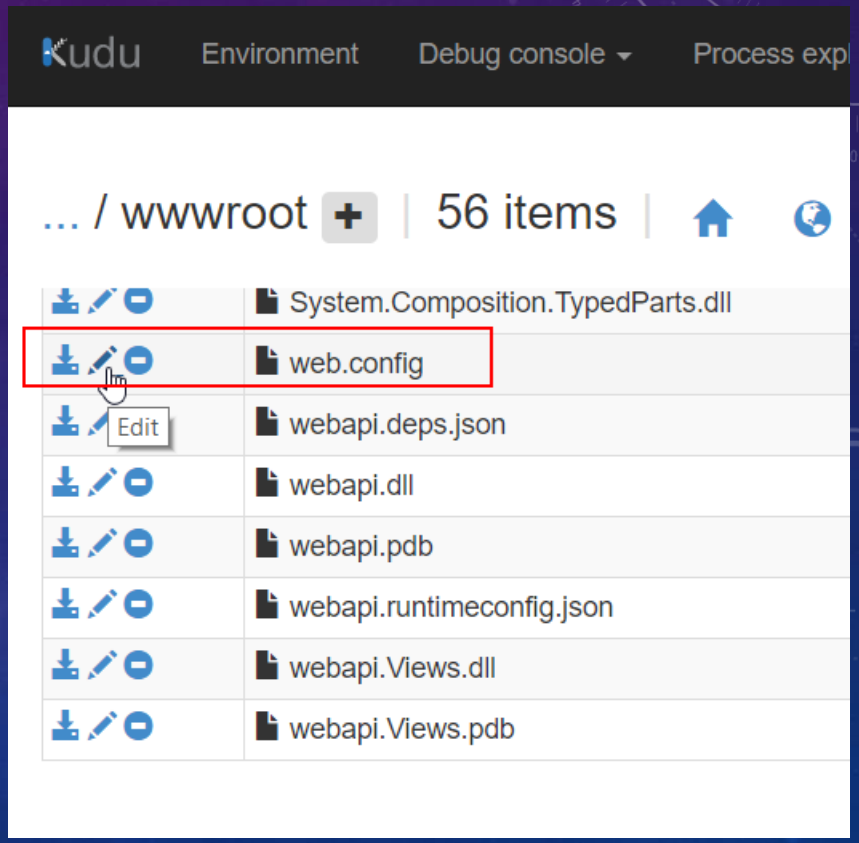
- 確認WebApp這兩個extension有安裝且版本和適用系統(x86/x64)有符合：





Deployment – Azure WebApp佈署troubleshoot

- 在Kudu上打開Visual Studio執行佈署或dotnet publish所產生的web.config：(D:/home/site/wwwroot/web.config)



Deployment – Azure WebApp佈署troubleshoot

- 啟用stdoutLogEnabled，並設定log檔位置(不能在D:\home\site目錄之外)：

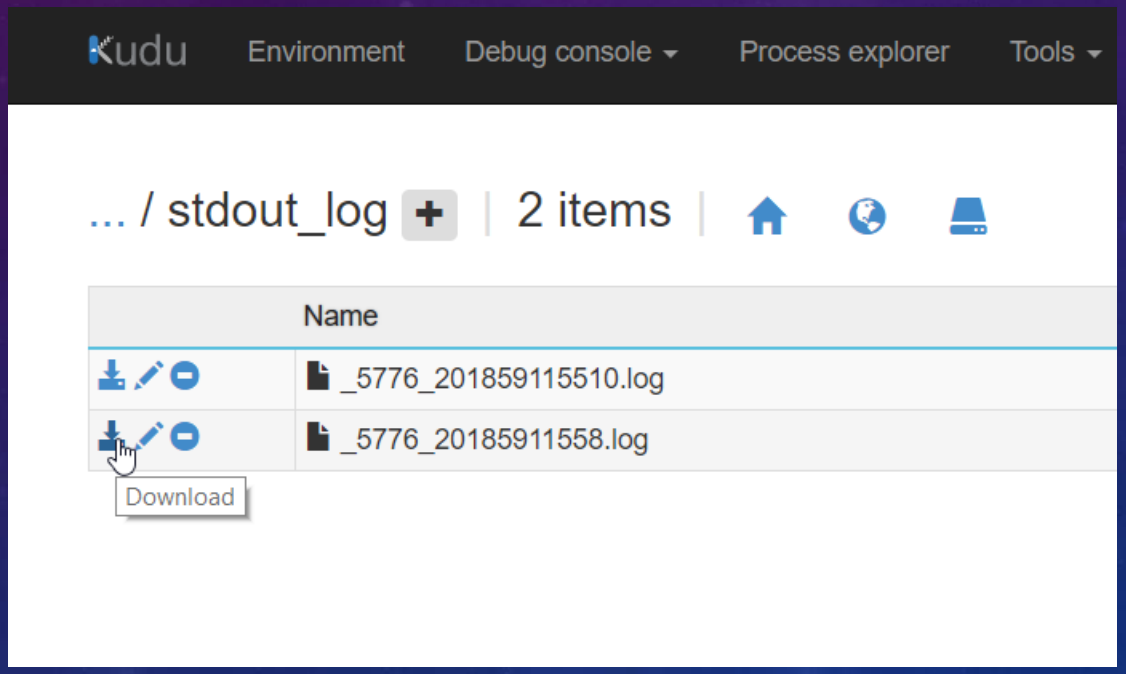
Kudu Environment Debug console Process explorer Tools Site extensions

Save Cancel web.config Help

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <configuration>
3    <system.webServer>
4      <handlers>
5        <add name="aspNetCore" path="*" verb="*" modules="AspNetCoreModule" resourceType="Unspecified" />
6      </handlers>
7      <aspNetCore processPath="dotnet" arguments=".\webapi.dll" stdoutLogEnabled="true" stdoutLogFile="..\stdout_log\" />
8    </system.webServer>
9  </configuration>
10 <!--ProjectGuid: [redacted]-->
```

Deployment – Azure WebApp佈署troubleshoot

- 再連網站一次，就會有Log檔案產生：



Deployment – Azure WebApp佈署troubleshoot

- 看Log檔好過觀落陰：

_5776_201859115510.log - Visual Studio Code

File Edit Selection View Go Debug Tasks Help

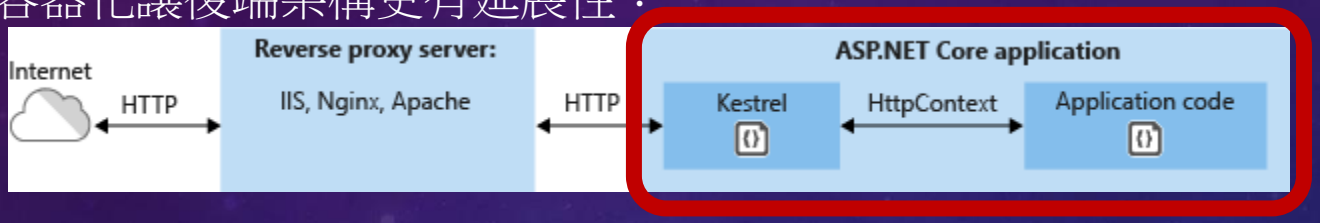
_5776_20185911558.log | _5776_201859115510.log

```
It was not possible to find any compatible framework version
The specified framework 'Microsoft.AspNetCore.App', version '2.1.0-rc1-final' was not found.
  - Check application dependencies and target a framework version installed at:
      D:\home\SiteExtensions\AspNetCoreRuntime.2.1.x64\
  - Installing .NET Core prerequisites might help resolve this problem:
      http://go.microsoft.com/fwlink/?LinkID=798306&clcid=0x409
  - The .NET Core framework and SDK can be installed from:
      https://aka.ms/dotnet-download
  - The following versions are installed:
      2.1.0-preview2-final at [D:\home\SiteExtensions\AspNetCoreRuntime.2.1.x64\shared\Microsoft.AspNetCore.App]
```

✓ 建議佈署至Azure Web App要勾選把先前檔案清除掉的選項，否則很容易又是吃到舊的assembly dll而造成IIS Http Error 502.5錯誤

Deployment – 容器化(Dockerize)佈署

- 容器化讓後端架構更有延展性：



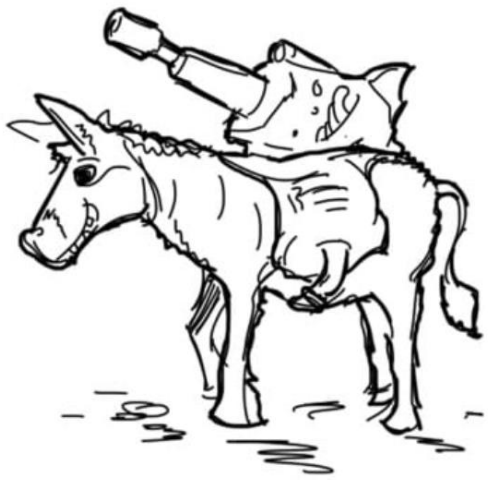
- 目前不同平台不同用途的Docker image不一樣，將來會用Docker的Multi-stage build技術來將docker image一致化占用空間更縮小：https://github.com/dotnet/announcements/issues/14
- Docker image各個不同tag用途的說明：https://github.com/dotnet/dotnet-docker#image-variants

✓ *.json, .xml設定檔、HTTPS憑證，建議配合docker-compose的yml設定檔，用volume mount進docker container的方式，而不是一起被docker build進去。https://blog.matjanowski.pl/2017/11/18/managing-options-and-secrets-in-net-core-and-docker/

Q & A

If programming languages were "weapons"

https://9gag.com/gag/anXEbe0/if-programming-languages-were-weapons



C# is a powerful laser rifle strapped to a donkey, when taken off the donkey the laser doesn't seem to work as well.

BONUS:
如何找ASP.NET Core官方Nuget套件的專案程式碼

1. Clone ASP.NET發行用的github檔案庫，這是git submodule的wrapper repo：
   https://github.com/aspnet/Universe

```
D:\repo\aspnet
> git clone --recursive https://github.com/aspnet/Universe.git -b release/2.1
Cloning into 'Universe'...
remote: Counting objects: 12847, done.
remote: Compressing objects: 100% (45/45), done.
remote: Total 12847 (delta 39), reused 49 (delta 23), pack-reused 12779
Receiving objects: 100% (12847/12847), 19.07 MiB | 1.11 MiB/s, done.
Resolving deltas: 100% (8430/8430), done.
Submodule 'modules/AADIntegration' (https://github.com/aspnet/AADIntegration.git) registered for
path 'modules/AADIntegration'
Submodule 'modules/Antiforgery' (https://github.com/aspnet/Antiforgery.git) registered for path '
modules/Antiforgery'
Submodule 'modules/AuthSamples' (https://github.com/aspnet/AuthSamples.git) registered for path '
modules/AuthSamples'
Submodule 'modules/AzureIntegration' (https://github.com/aspnet/AzureIntegration.git) registered
for path 'modules/AzureIntegration'
Submodule 'modules/BasicMiddleware' (https://github.com/aspnet/BasicMiddleware.git) registered fo
r path 'modules/BasicMiddleware'
Submodule 'modules/BrowserLink' (https://github.com/aspnet/BrowserLink.git) registered for path '
modules/BrowserLink'
```

~~然後可以試著用它根目錄的build.cmd全部一口氣build起來，會很久XD~~

2. 在modules目錄底下搜尋和套件全名相同名稱的子目錄：

https://www.nuget.org/packages/Microsoft.AspNetCore.Http.Extensions/2.1.0-rc1-final

nuget Packages Upload Statistics Documentation Downloads Blog Sign in

Search for packages...

.NET

Microsoft.AspNetCore.Http.Extensions

2.1.0-rc1-final

ASP.NET Core common extension methods for HTTP abstractions, HTTP headers, HTTP request/response, and session state.

This is a prerelease vers

Package Manager .NET CLI

PM> Install-Package Micros

Info

last updated 18 days ago

https://asp.net/

Dependenc

Version Hist

Microsoft.AspNetCore.Http.Extensions - search-ms:displayname=Search%20Results%20in%2...

File Home Share View Search Search Tools

Search Results in modules

Microsoft.AspNetCore.Http.Extensions

Quick access

OneDrive

This PC

Network

Microsoft.AspNetCore.Http.Extensions.Tests
Date modified: 5/25/2018 1:29 PM

Extensions
Date modified: 5/25/2018 1:29 PM

Microsoft.AspNetCore.Http.Extensions
Date modified: 5/25/2018 1:29 PM

Extensions
Date modified: 5/25/2018 1:29 PM

AutobahnTests.cs
D:\repo\aspnet\Universe\modules\WebSockets\test\Microsoft.AspNetC...
Date modified: 5/25/2018 1:30 PM
Size: 5.35 KB

3. 如果有和該目錄同名的.csproj檔，該專案就是那個官方套件的原始碼，可用來建置/修改自己debug版的Nuget套件使用。