
Host by
STUDY4
.NET Conf 2019
探索 .NET 新世界
.NET
dotnet new

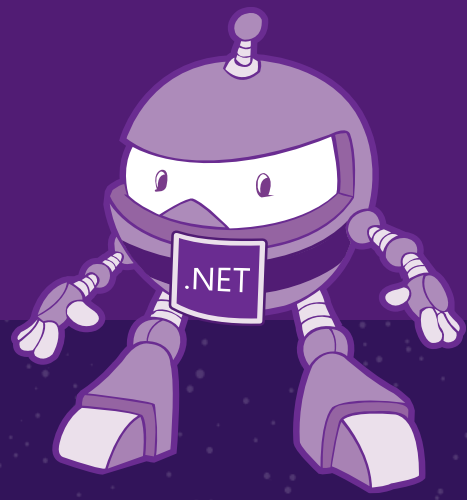
你不可不知的 ASP.NET Core 3 全新功能探索

多奇數位創意有限公司

技術總監 黃保翕 (Will 保哥)

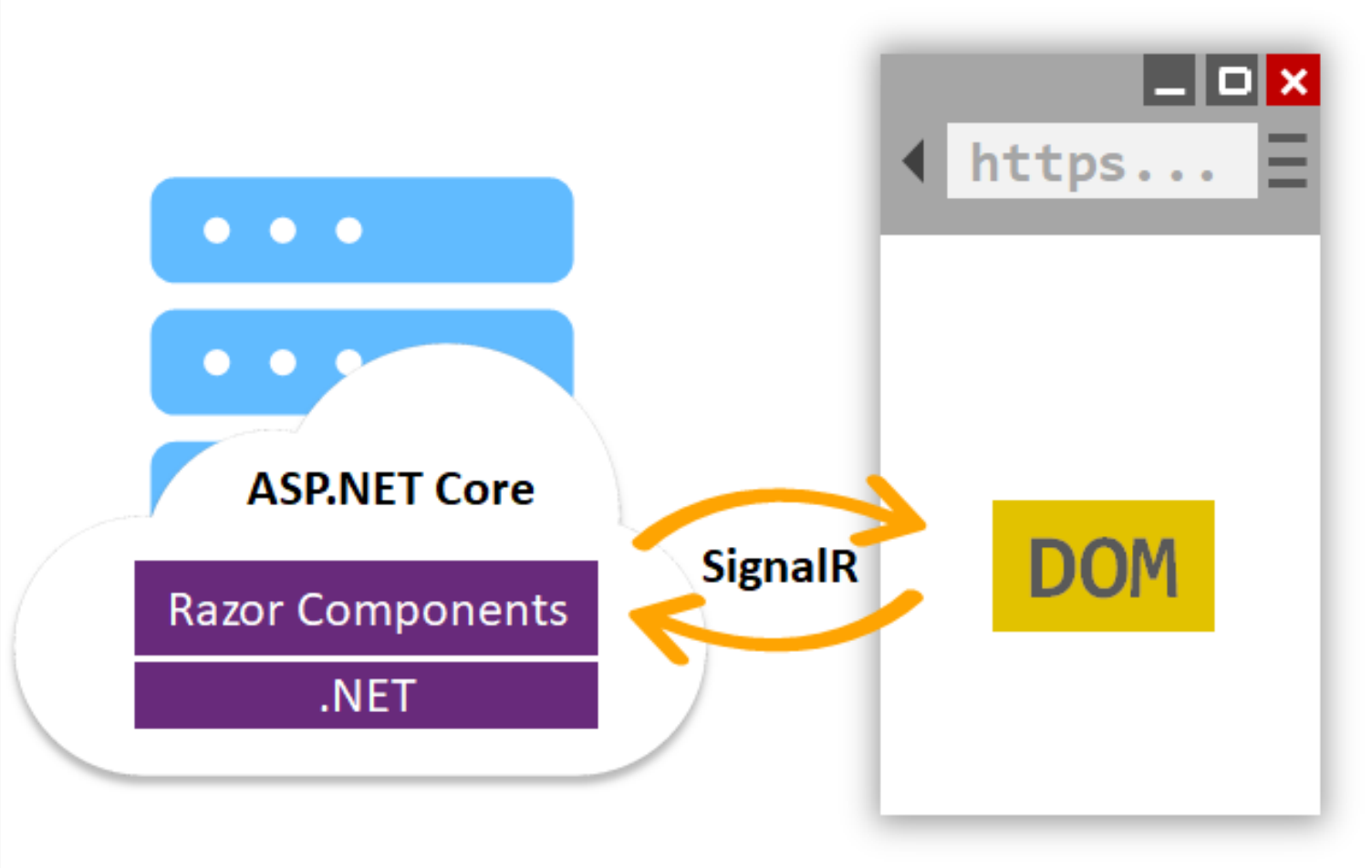
部落格：http://blog.miniasp.com/

Microsoft
Regional Director



Blazor Server

Blazor Server 運作架構



採用 Blazor Server 的理由

- 跨平台 (Cross-platform)
- 單一頁面應用程式 (SPA)
- 搜尋引擎最佳化 (SEO) (支援伺服器渲染) (SSR)
- 可以跟 MVC 與 Razor Pages 放在同一個專案
- 極低的 TTFB 與極佳的瀏覽體驗
- 不支援**離線操作**情境

Blazor 執行效能

- Standard D1 v2 instance on Azure (1 vCPU, 3.5 GB memory)
  - 可承受 5,000 同時連線數
- Standard D3 V2 instance on Azure (4 vCPU, 14GB memory)
  - 可承受 20,000 同時連線數

Blazor 的下一步？(.NET 5) (Nov 2020)

- Blazor PWAs
  - 增加離線功能、推播通知、作業系統整合 (釘選, 桌面捷徑, 開始選單)
- Blazor Hybrid
  - Blazor with Electron
- Blazor Native
  - 實現不同的 Render Engine (類似 Xamarin 與 React Native)

gRPC

什麼是 gRPC (https://grpc.io/)

- gRPC 是一套跟程式語言無關、高效能的 RPC 框架
  (RPC = Remote Procedure Call)
- 提供**以合約為主**的 API 開發模式 (Protocol Buffers)
- Introduction to gRPC on .NET Core

ASP.NET Core 3.0 支援的 gRPC 套件

- Grpc.AspNetCore
  - An ASP.NET Core framework for hosting gRPC services. gRPC on ASP.NET Core integrates with standard ASP.NET Core features like logging, dependency injection (DI), authentication, and authorization.
- Grpc.Net.Client
  - A gRPC client for .NET Core that builds upon the familiar HttpClient.
- Grpc.Net.ClientFactory
  - gRPC client integration with HttpClientFactory.

快速上手 gRPC

- [Tutorial: Create a gRPC client and server in ASP.NET Core]

```
# gRPC Server

dotnet new grpc -n g1
cd g1
dotnet run
```

```
# gRPC Client

dotnet new console -n c1
cd c1
dotnet add package Grpc.Net.Client
dotnet add package Google.Protobuf
dotnet add package Grpc.Tools
# 複製 gRPC Server 的 Protos 目錄過來
# 加入 <Protobuf> 項目到 *.csproj 專案檔中
<Protobuf Include="Protos/name.proto"
          GrpcServices="Client" />
# 建置專案 ( dotnet build )
# 修改 Program.cs (gprc-sample)
dotnet run
```

SignalR

主要改變

- 改用 System.Text.Json 序列化/反序列化 JSON 訊息
- JavaScript 與 .NET Clients for SignalR 支援**斷線自動重連**機制

```
const connection = new signalR.HubConnectionBuilder()
    .withUrl("/chatHub")
    .withAutomaticReconnect().build();
```

- 支援**斷路器**設計 (Circuit Breaker)

```
const connection = new signalR.HubConnectionBuilder()
    .withUrl("/chatHub")
    .withAutomaticReconnect([0, 3000, 5000, 10000, 15000])
    .build();
```

Tutorial: Get started with ASP.NET Core SignalR

- dotnet new webapp -o SignalRChat
- code -r SignalRChat
- # Install 3rd-party JS libraries
  - dotnet tool install -g Microsoft.Web.LibraryManager.Cli
  - libman install @microsoft/signalr@latest -p unpkg -d wwwroot/js/signalr --files dist/browser/signalr.js --files dist/browser/signalr.min.js
- # Hubs/ChatHub.cs
  - signalr-chat
- # Startup.cs
  - using SignalRChat.Hubs;
  - services.AddSignalR();
  - endpoints.MapHub<ChatHub>("/chatHub");
- # Pages/Index.cshtml
  - signalr-chat
- # wwwroot/js/chat.js
  - signalr-chat
- dotnet run

System.Text.Json

為什麼要重新發明輪子

- 提供高效能可解析 JSON 文件的 APIs (.NET Core 限定)
  - 主要使用 Span<T> 來提升效能
  - 比起 Json.NET 擁有 1.3 ~ 5 倍的效能提升 (未來還能改善更多)
- 最佳化 UTF-8 文字編碼處理
  - 直接跳過 .NET 內建字串的 UTF-16 轉換程序
- 直接改善 Json.NET 會造成更多破壞性更新
  - 從 ASP.NET Core 3 中移除所有跟 Json.NET 的相依性
  - 提供額外套件可以整合 Json.NET 回到 ASP.NET Core 之中

JSON 序列化

```
using System.Text.Json;
using System.Text.Json.Serialization;
```

```
class WeatherForecast
{
    public DateTimeOffset Date { get; set; }
    public int TemperatureC { get; set; }
    public string Summary { get; set; }
}
```

```
string Serialize(WeatherForecast value)
{
    return JsonSerializer.ToString<WeatherForecast>(value);
}
```

JSON 反序列化

```
// {
//      "Date": "2013-01-20T00:00:00Z",
//      "TemperatureC": 42,
//      "Summary": "Typical summer in Seattle. Not.",
// }
WeatherForecast Deserialize(string json)
{
    var options = new JsonSerializerOptions
    {
        AllowTrailingCommas = true
    };

    return JsonSerializer.Parse<WeatherForecast>(json, options);
}
```

透過屬性控制序列化/反序列化行為

```
class WeatherForecast
{
    public DateTimeOffset Date { get; set; }

    // Always in Celsius.
    [JsonPropertyName("temp")]
    public int TemperatureC { get; set; }

    public string Summary { get; set; }

    // Don't serialize this property.
    [JsonIgnore]
    public bool IsHot => TemperatureC >= 30;
}
```

支援完整的 DOM 物件結構 (JsonDocument)

- 有時候你並不想解析完整的 JSON 文件之後才存取裡面的資料

```
using (JsonDocument document = JsonDocument.Parse(json, options))
{
    foreach (JsonElement element in document.RootElement.EnumerateArray())
    {
        DateTimeOffset date = element.GetProperty("date").GetDateTimeOffset();

        if (date.DayOfWeek == DayOfWeek.Monday)
        {
            int temp = element.GetProperty("temp").GetInt32();
            sumOfAllTemperatures += temp;
            count++;
        }
    }
}
```

推薦連結

- Try the new System.Text.Json APIs | .NET Blog
  - 詳細解說 System.Text.Json 的運作原理與使用方式
- Try the new System.Text.Json APIs! | On .NET | Channel 9
  - 簡短 19 分鐘的技術分享
- System.Text.Json 命名空間
  - 完整的 API 文件
- The future of JSON in .NET Core 3.0 #33115

Kestrel

關於 Kestrel 的變更

- HTTPS 加密連線預設支援 HTTP/2 通訊協定
- 每個 HTTP 要求的 EventCounter 現在會送出以下事件
  (Windows 事件追蹤 (ETW) 的 EventSource) (EventSource provider)
  - requests-per-second
  - total-requests
  - current-requests
  - failed-requests
- Migrate from ASP.NET Core 2.2 to 3.0 | Kestrel

身分識別與授權

Web API 與 SPAs 可搭配 IdentityServer4 整合

- 更好的整合認證授權機制與完整的 SPA 專案範本
  - `dotnet new angular -o ng8 -au Individual`
  - `dotnet new react -o react1 -au Individual`
- 完整文件
  - Authentication and authorization for SPAs
  - Use the Angular project template with ASP.NET Core
  - Use the React project template with ASP.NET Core
  - Scaffold Identity in ASP.NET Core projects

支援憑證認證 (Certificate authentication)

- 支援自簽憑證、憑證撤銷檢查、檢查憑證是否正確設定 Flags
  - Configure certificate authentication in ASP.NET Core

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddAuthentication(
        CertificateAuthenticationDefaults.AuthenticationScheme)
            .AddCertificate();
}
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    app.UseAuthentication();
}
```

Windows Authentication 支援 Linux 與 macOS

- ASP.NET Core 3.0 新版的 Kestrel 支援多種新的認證方式
  - Negotiate
  - Kerberos
  - NTLM on Windows, Linux, and macOS
- 記得安裝 Microsoft.AspNetCore.Authentication.Negotiate 套件

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddAuthentication(NegotiateDefaults.AuthenticationScheme)
        .AddNegotiate();
}
```

MVC 與 Razor Pages

端點路由機制 (Endpoint routing)

- ASP.NET Core 2.x 使用 **app.UseMvc()** 與 **app.UseSignalR()**
- ASP.NET Core 3.x 建議的設定順序為

```
app.UseStaticFiles();       // 必須在 UseRouting() 之前被呼叫
app.UseRouting();
app.UseCors();              // 必須在所有會用到 CORS 的中介層之前被呼叫
app.UseAuthentication();
app.UseAuthorization();
app.UseEndpoints(endpoints => {
    endpoints.MapControllers();
});
```

健康檢查服務 (Health Checks)

- 改用端點路由機制設定**健康檢查服務** (Health checks)

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddHealthChecks();
}
public void Configure(IApplicationBuilder app)
{
    app.UseRouting();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapHealthChecks("/health");
    });
}
```

全新的 Razor 指令 (directives)

- @attribute
  - 其實所有 Razor 頁面都會動態產生一個類別 (Class)
  - 你現在可以透過 @attribute 標示一個 Attribute 屬性到此類別
  - 例如：`@attribute [Authorize]`
- @implements
  - 你可以透過 @implements 宣告該 Razor 類別是否要實作特定介面
  - 例如：`@implements IDisposable`

.NET Core 專案範本變更 (MVC, Razor Pages)

- 移除 GDPR 所需的 Cookie Consent UI
  - 有需要的請自行安裝設定：EU GDPR support in ASP.NET Core
- 範本中會用到的第三方 JS、CSS 與其他檔案都改從 Local 版本
  - 早期會直接使用 CDN 網址，但許多企業對外基本上是斷線的！
- Angular 專案範本升級到 Angular v8.0.6 版本
- Razor class library (RCL) 預設改採用 Razor component (Blazor)
  - dotnet new razorclasslib -n r1
  - dotnet new razorclasslib -n r2 --support-pages-and-views (舊版格式)

從 ASP.NET Core 2 升級 3.0

廢棄 WebHostBuilder 改用 .NET Generic Host

- 為了讓 ASP.NET Core 可以更好的整合不同的應用情境
  - ASP.NET Core / Worker / gRPC service / Blazor Server
- 預設環境變數名稱前置詞變更
  - ASP.NET Core 2.x：`ASPNETCORE_` (WebHost)
  - ASP.NET Core 3.x：`DOTNET_` (GenericHost)
    - 請參考 CreateDefaultBuilder() 原始碼
- Startup 類別的建構式只支援三種型別 DI 注入 (#353)
  - `IHostEnvironment`
  - `IConfiguration`
  - `IWebHostEnvironment`

使用 HostBuilder 取代 WebHostBuilder

- ASP.NET Core 2.x

```
public class Program
{
    public static void Main(string[] args)
    {
        CreateWebHostBuilder(args).Build().Run();
    }

    public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
            .UseStartup<Startup>();
}
```

用 HostBuilder 取代 WebHostBuilder

- ASP.NET Core 3.x

```
public class Program
{
    public static void Main(string[] args)
    {
        CreateHostBuilder(args).Build().Run();
    }

    public static IHostBuilder CreateHostBuilder(string[] args) =>
        Host.CreateDefaultBuilder(args)
            .ConfigureWebHostDefaults(webBuilder =>
            {
                webBuilder.UseStartup<Startup>();
            });
}
```

ASP.NET Core 2.2 的路由定義

```
public void Configure(IApplicationBuilder app)
{
  app.UseStaticFiles();

  app.UseCors();

  app.UseAuthentication();

  app.UseSignalR(hubs =>
  {
    hubs.MapHub<ChatHub>("/chat");
  });

  app.UseMvc(routes =>
  {
    routes.MapRoute("default", "{controller=Home}/{action=Index}/{id?}");
  });
}
```

ASP.NET Core 3.0 的路由定義

```
public void Configure(IApplicationBuilder app)
{
  app.UseStaticFiles();

  app.UseRouting();

  app.UseCors();

  app.UseAuthentication();
  app.UseAuthorization();

  app.UseEndpoints(endpoints =>
  {
    endpoints.MapHub<ChatHub>("/chat");
    endpoints.MapControllerRoute("default", "{controller=Home}/{action=Index}/{id?}");
  });
}
```

從 ASP.NET Core 3.0 移除的套件

- Newtonsoft.Json (Json.NET)
  - 改回 Newtonsoft.Json-based 的 JSON 格式可參考：
    Add Newtonsoft.Json-based JSON format support
- Entity Framework Core
  - **dotnet tool install --global dotnet-ef**
  - dotnet add package Microsoft.EntityFrameworkCore.Design
  - dotnet add package Microsoft.EntityFrameworkCore.Sqlite
  - dotnet add package Microsoft.EntityFrameworkCore.SqlServer
  - dotnet add package Microsoft.EntityFrameworkCore.InMemory
  - dotnet add package Oracle.EntityFrameworkCore

相關連結

- ASP.NET Documentation
- Introduction to ASP.NET Core
- What's new in ASP.NET Core 3.0 | Microsoft Docs
- Migrate from ASP.NET Core 2.2 to 3.0 | Microsoft Docs
- .NET Core Extension Pack - Visual Studio Marketplace
- https://github.com/doggy8088/DotNetConf2019Demo

聯絡資訊

The Will Will Web

網路世界的學習心得與技術分享

http://blog.miniasp.com/

Facebook

Will 保哥的技術交流中心

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

Twitter

https://twitter.com/Will_Huang

特別感謝

Microsoft® MVP Most Valuable Professional

SEA 台灣軟體工程學會 Taiwan

R-Ladies Taipei

TAIWAN

91APP

多奇·數位創意

Wishing-Soft

KKTIX

Geek & Nerd Studios

以及各位參與活動的你們

STUDY4.TW 為學習而生

.NET Conf