

# 第76期-从 ASP.NET Core 2.2 迁移到 3.0 版本

2019年12月12日 10:56

## 环境准备与先决条件

- Visual Studio 2019 与 ASP.NET 和 Web 开发 工作负载
- .NET Core 3.0 SDK 或更高版本

## 更新项目文件

### 更新目标框架

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>netcoreapp3.0</TargetFramework>
  </PropertyGroup>
</Project>
```

### 删除过时的包引用

#### 在 <PropertyGroup> 中:

将 TFM 更新为 netcoreapp3.0

删除 <AspNetCoreHostingModel> 元素。有关详细信息，请参阅本文档中的进程内承载模型。

#### 在 <ItemGroup> 中:

删除 Microsoft.AspNetCore.App。有关详细信息，请参阅本文档中的框架参考。

删除了 Microsoft.AspNetCore.Razor.Design，并且不再生成以下包列表。

**总之，只要有微软自带的包全部删除就可以了。**

## 使用 Docker 的依赖框架的生成

```
{
  "runtimeOptions": {
    "tfm": "netcoreapp3.0",
    "framework": {
      "name": "Microsoft.AspNetCore.App",
      "version": "3.0.0"
    }
  }
}
```

```

    },
    "configProperties": {
        "System.GC.Server": true
    }
}
}

```

将使用的第三方 NuGet 包升级到最新

## 更改启动项

```

Diff - Startup.cs;HEAD vs. Startup.cs
Web1.csproj  Startup.cs  Web1
Startup.cs;HEAD
Miscellaneous Files  Web1.Startup  Startup(IConfiguration configurat
1  using Microsoft.AspNetCore.Builder;
2  using Microsoft.AspNetCore.Hosting;
3  using Microsoft.AspNetCore.Mvc;
4  using Microsoft.Extensions.Configuration;
5  using Microsoft.Extensions.DependencyInjection;
6  //////////////////////////////////////////////////
7  namespace Web1
8  {
9      public class Startup
10     {
11         public Startup(IConfiguration configuration)
12         {
13             Configuration = configuration;
14         }
15
16         public IConfiguration Configuration { get; }
17
18         public void ConfigureServices(IServiceCollection services)
19         {
20             services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2)
21         }
22
23         public void Configure(IApplicationBuilder app, IHostingEnvironment env)
24         {
25             if (env.IsDevelopment())
26             {
27                 app.UseDeveloperExceptionPage();
28             }
29             else
30             {
31                 app.UseExceptionHandler("/Error");
32                 app.UseHsts();
33             }
34
35             app.UseHttpsRedirection();
36             app.UseStaticFiles();
37
38             app.UseMvc();
39             //////////////////////////////////////////////////
40         }
41     }

```

```

Startup.cs
Web1 Web1.Startup Startup(IConfiguration con
1 using Microsoft.AspNetCore.Builder;
2 using Microsoft.AspNetCore.Hosting;
3 using Microsoft.Extensions.Configuration;
4 using Microsoft.Extensions.DependencyInjection;
5 using Microsoft.Extensions.Hosting;
6
7 namespace Web1
8 {
9     public class Startup
10    {
11        public Startup(IConfiguration configuration)
12        {
13            Configuration = configuration;
14        }
15
16        public IConfiguration Configuration { get; }
17
18        public void ConfigureServices(IServiceCollection services)
19        {
20            services.AddRazorPages();
21        }
22
23        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
24        {
25            if (env.IsDevelopment())
26            {
27                app.UseDeveloperExceptionPage();
28            }
29            else
30            {
31                app.UseExceptionHandler("/Error");
32                app.UseHsts();
33            }
34
35            app.UseHttpsRedirection();
36            app.UseStaticFiles();
37
38            app.UseRouting();
39
40            app.UseAuthorization();
41
42            app.UseEndpoints(endpoints =>
43            {
44                endpoints.MapRazorPages();
45            });
46        }
47    }
48 }
49

```

## 分析器支持

```

<PropertyGroup>
<IncludeOpenAPIAnalyzers>true</IncludeOpenAPIAnalyzers>
</PropertyGroup>

```

## Razor 类库支持

```
<PropertyGroup>
    <AddRazorSupportForMvc>true</AddRazorSupportForMvc>
</PropertyGroup>
```

## Json.NET 支持

作为 ASP.NET Core 共享框架的工作的一部分，已从 ASP.NET Core 共享框架中删除 Json.NET 框架。

ASP.NET Core 的默认值现在是 .NET Core 3.0 中的新系统。如果可能，请考虑使用 System.Text.Json。它是高性能的，不需要额外的库依赖项。但是，由于 System.Text.Json 是新的，因此它当前可能缺少应用需要的功能。

```
services.AddMvc().AddNewtonsoftJson();
services.AddControllers().AddNewtonsoftJson();

services.AddMvc()
    .AddNewtonsoftJson(options =>
        options.SerializerSettings.ContractResolver =
            new CamelCasePropertyNamesContractResolver());
```

## MVC 服务注册

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllers();
}
```

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
}
```

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddRazorPages();
}
```

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
    services.AddRazorPages();
}
```

## 路由启动代码

如果应用调用 UseMvc 或 UseSignalR，则将应用迁移到终结点路由。

```
public void Configure(IApplicationBuilder app)
{
    app.UseStaticFiles();
    app.UseAuthentication();
    app.UseSignalR(hubs =>
    {
        hubs.MapHub<ChatHub>("/chat");
    });
    app.UseMvc(routes =>
    {
        routes.MapRoute("default", "{controller=Home}/{action=Index}/{id?}");
    });
}
```

```
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?}");
    });
}
```

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

```
public void Configure(IApplicationBuilder app)
{
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapRazorPages();
    });
}
```

**注意 app.UseRouting(); 必须出现在最前。**

## 使用不带终结点路由的 MVC

```
services.AddMvc(options => options.EnableEndpointRouting = false);
services.AddControllers(options => options.EnableEndpointRouting = false);
services.AddControllersWithViews(options => options.EnableEndpointRouting = false);
```

```
services.AddRazorPages().AddMvcOptions(options => options.EnableEndpointRouting = false);
```

## HostBuilder 替换 WebHostBuilder

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

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

```
// requires using Microsoft.AspNetCore.Hosting;
// requires using Microsoft.Extensions.Hosting;
```

```
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>();
            });
}
```

IWebHostBuilder 保留在3.0，并且是前面的代码示例中所见 webBuilder 的类型。未来版本中将弃用 WebHostBuilder，并将其替换为 HostBuilder。

## SignalR 序列化库回退

```
services.AddSignalR(...)
    .AddJsonProtocol(options =>
    {
        options.PayloadSerializerOptions.WriteIndented = false;
    })

new HubConnectionBuilder()
    .WithUrl("/chatHub")
```

```
.AddJsonProtocol(options =>
{
    options.PayloadSerializerOptions.WriteIndented = false;
})
.Build();

services.AddSignalR()
    .AddNewtonsoftJsonProtocol(...);
```