## **Startup 类**

ASP.NET Core 应用使用 Startup 类作为启动类，按照约定命名为 Startup。

当应用启动时，Startup的 ConfigureServices 和 Configure方法被运行

public class Startup

{

// Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

...

}

// Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app)

{

...

}

}

通过 [WebHostBuilderExtensions](https://docs.microsoft.com/zh-cn/dotnet/api/Microsoft.AspNetCore.Hosting.WebHostBuilderExtensions)、[UseStartup<TStartup>](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.hosting.webhostbuilderextensions.usestartup" \l "Microsoft_AspNetCore_Hosting_WebHostBuilderExtensions_UseStartup__1_Microsoft_AspNetCore_Hosting_IWebHostBuilder_) 方法指定 Startup 类：

public class Program

{

public static void Main(string[] args)

{

CreateWebHostBuilder(args).Build().Run();

}

public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>

WebHost

.CreateDefaultBuilder(args)

.UseStartup<Startup>();

}

在 Startup 类构造函数中可以注入如下类：

* [IHostingEnvironment](https://docs.microsoft.com/zh-cn/dotnet/api/Microsoft.AspNetCore.Hosting.IHostingEnvironment) 以按环境配置服务。
* [IConfiguration](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.extensions.configuration.iconfiguration) 以在启动过程中配置应用。

## **ConfigureServices 方法**

将服务添加到服务容器，使其在应用和 Configure 方法中可用。

public void ConfigureServices(IServiceCollection services)

{

// Add framework services.

services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

services.AddIdentity<ApplicationUser, IdentityRole>()

.AddEntityFrameworkStores<ApplicationDbContext>()

.AddDefaultTokenProviders();

services.AddMvc();

// Add application services.

services.AddTransient<IEmailSender, AuthMessageSender>();

services.AddTransient<ISmsSender, AuthMessageSender>();

}

[IServiceCollection](https://docs.microsoft.com/zh-cn/dotnet/api/Microsoft.Extensions.DependencyInjection.IServiceCollection) 上有 Add[Service] 扩展方法。

### **ASP.NET Core MVC 的 SetCompatibilityVersion**

SetCompatibilityVersion 方法允许应用选择加入或退出 ASP.NET MVC Core 2.1+ 中引入的潜在中断行为变更。

以下代码将兼容模式设置为 ASP.NET Core 2.1：

public void ConfigureServices(IServiceCollection services)

{

services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);

}

## **Configure 方法**

[Configure](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.hosting.startupbase.configure) 方法用于指定应用响应 HTTP 请求的方式。 可通过将[中间件](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/middleware/index?view=aspnetcore-2.1)组件添加到 [IApplicationBuilder](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.builder.iapplicationbuilder)实例来配置请求管道。

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

app.UseBrowserLink();

}

else

{

app.UseExceptionHandler("/Error");

}

app.UseStaticFiles();

app.UseMvc(routes =>

{

routes.MapRoute(

name: "default",

template: "{controller}/{action=Index}/{id?}");

});

}

每个 Use 扩展方法将中间件组件添加到请求管道。

## **Startup 筛选器**

[IStartupFilter](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.hosting.istartupfilter) 相当于IStartup，但[IStartupFilter](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.hosting.istartupfilter) 可以对IStartup生成的中间件进行过滤

public class RequestSetOptionsStartupFilter : IStartupFilter

{

public Action<IApplicationBuilder> Configure(Action<IApplicationBuilder> next)

{

return builder =>

{

builder.UseMiddleware<RequestSetOptionsMiddleware>();

next(builder);

};

}

}

在 ConfigureServices 的服务容器中注册 IStartupFilter：

public void ConfigureServices(IServiceCollection services)

{

services.AddTransient<IStartupFilter, RequestSetOptionsStartupFilter>();

services.AddMvc();

}