可以注册 [IHttpClientFactory](https://docs.microsoft.com/zh-cn/dotnet/api/system.net.http.ihttpclientfactory) 并将其用于配置和创建应用中的 [HttpClient](https://docs.microsoft.com/zh-cn/dotnet/api/system.net.http.httpclient) 实例

## **系统必备**

面向.NET Framework 的项目要求安装 [Microsoft.Extensions.Http](https://www.nuget.org/packages/Microsoft.Extensions.Http/) NuGet 包。

面向 .NET Core 且引用 [Microsoft.AspNetCore.App 元包](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/metapackage-app?view=aspnetcore-2.1)的项目已经包括 Microsoft.Extensions.Http 包。

### **基本用法**

注册服务

services.AddHttpClient();

依赖注入

public class BasicUsageModel : PageModel {

    private readonly IHttpClientFactory \_clientFactory;

public IEnumerable<GitHubBranch> Branches { get; private set; }

public bool GetBranchesError { get; private set; }

    // 注入 IHttpClientFactory

    public BasicUsageModel (IHttpClientFactory clientFactory) {

        \_clientFactory = clientFactory;

}

    public async Task OnGet () {

        var request = new HttpRequestMessage (HttpMethod.Get, "https://api.github.com/repos/aspnet/docs/branches");

        request.Headers.Add ("Accept", "application/vnd.github.v3+json");

        request.Headers.Add ("User-Agent", "HttpClientFactory-Sample");

        // 生成 Http 客户端

        var client = \_clientFactory.CreateClient ();

        // 发送请求

        var response = await client.SendAsync (request);

        if (response.IsSuccessStatusCode) {

            Branches = await response.Content.ReadAsAsync<IEnumerable<GitHubBranch>> ();

        } else {

            GetBranchesError = true;

            Branches = Array.Empty<GitHubBranch> ();

        }

    }

}

### **命名客户端**

services.AddHttpClient ("github",

    c => {

        c.BaseAddress = new Uri ("https://api.github.com/");

        c.DefaultRequestHeaders.Add ("Accept", "application/vnd.github.v3+json");

        // Github API versioning

        c.DefaultRequestHeaders.Add ("User-Agent", "HttpClientFactory-Sample");

        // Github requires a user-agent

    }

);

// 创建命名客户端

var client = \_clientFactory.CreateClient("github");

## **出站请求中间件**

要创建处理程序，请定义一个派生自 DelegatingHandler 的类。 重写 SendAsync 方法，在将请求传递至管道中的下一个处理程序之前执行代码

public class ValidateHeaderHandler : DelegatingHandler {

    protected override async Task<HttpResponseMessage> SendAsync (HttpRequestMessage request,

        CancellationToken cancellationToken) {

        if (!request.Headers.Contains ("X-API-KEY")) {

            return new HttpResponseMessage (HttpStatusCode.BadRequest) {

                Content = new StringContent ("You must supply an API key header called X-API-KEY")

            };

        }

        return await base.SendAsync (request, cancellationToken);

    }

}

services.AddTransient<ValidateHeaderHandler> ();

services.AddHttpClient ("externalservice", c => {

        c.BaseAddress = new Uri ("https://localhost:5000/"); // assume this is an "external" service which requires an API KEY

    })

    .AddHttpMessageHandler<ValidateHeaderHandler> ();