包：Microsoft.Extensions.Http

**向IServiceCollention注册IHttpClientFactory**

Microsoft.Extensions.Http针对IServiceCollention提供了扩展方法AddHttpClient();

IHostBuilder builder = Host.CreateDefaultBuilder();

builder.ConfigureServices(services =>

{

    services.AddHttpClient();

});

**基本使用**

    public class MyService

    {

        private IHttpClientFactory \_httpClientFactory { get; }

        // 依赖注入IHttpClientFactory

        public MyService(IHttpClientFactory httpClientFactory) {

            this.\_httpClientFactory = httpClientFactory;

        }

        public async Task Fetch(CancellationToken cancellationToken)

        {

            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 = \_httpClientFactory.CreateClient();

            // 发送请求

            var response = await client.SendAsync(request);

            if (response.IsSuccessStatusCode)

            {

                string body = await response.Content.ReadAsStringAsync();

                Console.WriteLine("请求成功", body);

            }

            else

            {

                Console.WriteLine("请求失败");

            }

        }

    }

**命名客户端**

我们可以注册命名的HttpClient

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

    }

);

在创建时使用名称创建

// 创建命名客户端

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

**POST请求示例**

public async Task CreateItemAsync(Item item)

{

    using StringContent json = new(

        JsonSerializer.Serialize(item, DefaultJsonSerialization.Options),

        Encoding.UTF8,

        MediaTypeNames.Application.Json);

using HttpResponseMessage httpResponse = await \_httpClient.PostAsync("/api/items", json);

    httpResponse.EnsureSuccessStatusCode();

}

**PUT请求示例**

public async Task UpdateItemAsync(Item item)

{

    using StringContent json = new(

        JsonSerializer.Serialize(item, DefaultJsonSerialization.Options),

        Encoding.UTF8,

        MediaTypeNames.Application.Json);

using HttpResponseMessage httpResponse = await \_httpClient.PutAsync($"/api/items/{item.Id}", json);

    httpResponse.EnsureSuccessStatusCode();

}

**DELETE请求示例**

public async Task DeleteItemAsync(Guid id)

{

    using HttpResponseMessage httpResponse =

        await \_httpClient.DeleteAsync($"/api/items/{id}");

    httpResponse.EnsureSuccessStatusCode();

}