本章节将向读者讲解如何在不同的项目环境下，选择适合的方式来创建http声明接口的代理类。

1. 没有依赖注入的环境

1.1 使用HttpApiClient静态类(不推荐)

接口声明

```
public interface IMyWebApi : IHttpApi
{
    [HttpGet("user/{id}")]
    ITask<UserInfo> GetUserAsync(string id);
}
```

创建接口的全局实例

```
var httpApiConfig = new HttpApiConfig
{
   HttpHost = new Uri("http://localhost:9999/"),
   LoggerFactory = new LoggerFactory().AddConsole(),
};
this.myWebApi = HttpApiClient.Create<IMyWebApi>(httpApiConfig);


// 调用http请求代码
var user = await this.myWebApi.GetUserAsync("id001");
```

你应该尽量将得到的myWebApi保持为全局变量，多次请求里共用一个myWebApi实例。如果频繁地每次请求都创建和释放myWebApi，实际等同于短连接到服务器，客户端和服务器的性能都受到影响。但全局单例的myWebApi可能不遵循守DNS生存时间(TTL)设置，请求的域名指向的ip变化之后，会产生不正确的请求。

1.2 使用HttpApiFactory静态类

接口声明

```
public interface IMyWebApi : IHttpApi
{
   [HttpGet("user/{id}")]
   ITask<UserInfo> GetUserAsync(string id);
}
```

初始化代码（只能调用一次）

```
HttpApiFactory.Add<IMyWebApi>().ConfigureHttpApiConfig(c =>
{
   c.HttpHost = new Uri("http://localhost:9999/");
   c.LoggerFactory = new LoggerFactory().AddConsole();
   c.FormatOptions.DateTimeFormat = DateTimeFormats.ISO8601_WithMillisecond;
});
```

调用http请求代码

```
var myWebApi = HttpApiFactory.Create<IMyWebApi>();
var user = await myWebApi.GetUserAsync("id001");
```

使用HttpApiFactory的好处是在入口处只配置一次IMyWebApi，由HttpApiFactory自动接理IMyWebApi的生命周期管理。在使用中，不用处理myWebApi实例的释放（手动Dispose也不会释放），在一定的时间内都是获取到同一个myWebApi实例，当实例生命超过配置的周期时，自动被跟踪释放，并提供返回下一个一样配置的myWebApi实例。

2. 有依赖注入的环境

除了可以像上面使用HttpApiFactory静态类之外，WebApiClient还提供IHttpApiFactory<>和HttpApiFactory<>类型，很容易应用于各种有依赖注入的环境。在asp.net core的项目中，建议使用WebApiClient.Extensions项目简化配置，以下代码实际上是扩展项目的核心代码。

2.1 Asp.net MVC + Autofac

接口声明

```
public interface IMyWebApi : IHttpApi
{
    [HttpGet("user/{id}")]
    ITask<UserInfo> GetUserAsync(string id);
}
```

Global.asax.cs Application_Start

```
var builder = new ContainerBuilder();
builder.RegisterControllers(Assembly.GetExecutingAssembly()).PropertiesAutowired();

builder.Register(_ => new HttpApiFactory<IMyWebApi>()
   .ConfigureHttpApiConfig(c =>
   {
       c.HttpHost = new Uri("http://localhost:9999/");
       c.FormatOptions.DateTimeFormat = DateTimeFormats.ISO8601_WithMillisecond;
   }))
   .As<IHttpApiFactory<IMyWebApi>>()
   .SingleInstance();

builder.Register(c => c.Resolve<IHttpApiFactory<IMyWebApi>>().CreateHttpApi())
   .As<IMyWebApi>()
   .InstancePerHttpRequest();

DependencyResolver.SetResolver(new AutofacDependencyResolver(builder.Build()));
```

Controller

```
public class HomeController : Controller
{
    public IMyWebApi MyWebApi { get; set; }

    public async Task<ActionResult> Index()
    {
        var user = await this.MyWebApi.GetUserAsync("id001");
        return View(user);
    }
}
```

2.2 Asp.net Core

接口声明

```
public interface IMyWebApi : IHttpApi
{
    [HttpGet("user/{id}")]
    ITask<UserInfo> GetUserAsync(string id);
}
```

Startup.cs

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<IHttpApiFactory<IMyWebApi>, HttpApiFactory<IMyWebApi>>
(p =>
    {
        return new HttpApiFactory<IMyWebApi>().ConfigureHttpApiConfig(c =>
        {
            c.HttpHost = new Uri("http://localhost:9999/");
            c.LoggerFactory = p.GetRequiredService<ILoggerFactory>();
        });
    });

    services.AddTransient<IMyWebApi>(p =>
    {
        var factory = p.GetRequiredService<IHttpApiFactory<IMyWebApi>>();
        return factory.CreateHttpApi();
    });
}
```

Controller代码

```
public class HomeController : Controller
{
    public async Task<UserInfo> Index([FromServices]IMyWebApi myWebApi)
    {
        return await myWebApi.GetUserAsync("id001");
    }
}
```

2.3 Asp.net core + HttpClientFactory

接口声明

```
public interface IMyWebApi : IHttpApi
{
    [HttpGet("user/{id}")]
    ITask<UserInfo> GetUserAsync(string id);
}
```

Startup.cs配置依赖注入

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddHttpClient<IMyWebApi>().AddTypedClient<IMyWebApi>((client, p) =>
    {
        var httpApiConfig = new HttpApiConfig(client)
        {
            HttpHost = new Uri("http://localhost:9999/"),
            LoggerFactory = p.GetRequiredService<ILoggerFactory>()
        };
        return HttpApiClient.Create<IMyWebApi>(httpApiConfig);
    });
}
```

Controller代码

```
public class HomeController : Controller
{
    public async Task<UserInfo> Index([FromServices]IMyWebApi myWebApi)
    {
        return await myWebApi.GetUserAsync("id001");
    }
}
```