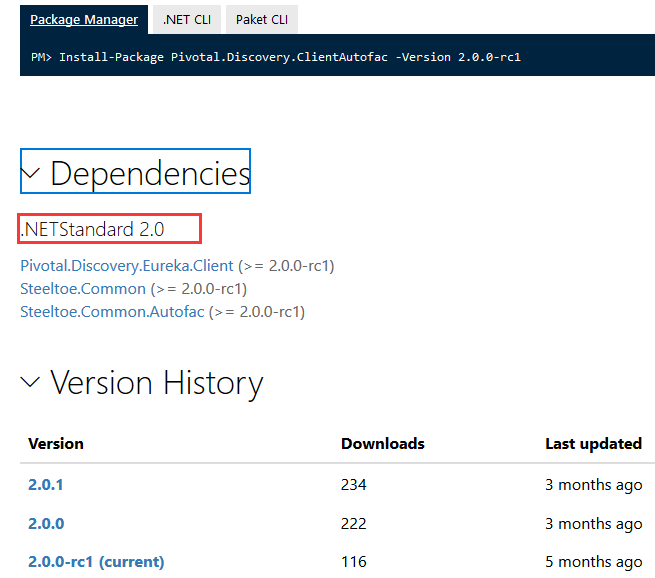
**AspNet4+SpringCloud微服务对接实例**

**1-新建基于.NET Framework 4.6.1目标框架的WebApi项目  
说明：Pivotal.Discovery.ClientAutofac最低版本为NETStandard2.0,所以在4.5.1版本需要对Autofac进行扩展，具体详见AspNet451项目源码**

**2-使用Nuget进行对Steeltoe包的引用**  
 PM> install-package Pivotal.Discovery.ClientAutofac

PM> install-package Autofac.WebApi2

1. **添加[appsettings.json]配置文件，如下：**

{

"Logging": {

"IncludeScopes": false,

"Debug": {

"LogLevel": {

"Default": "Warning"

}

},

"Console": {

"LogLevel": {

"Default": "Warning"

}

}

},

//注册到spring cloud的服务名

"spring": {

"application": {

"name": "Order"

}

},

//spring cloud的客户端配置

"eureka": {

"client": {

"serviceUrl": "http://localhost:8761/eureka/" //注册中心地址

//"serviceUrl": "http://192.168.2.118:8761/eureka/" //注册中心地址（测试环境）

},

"instance": {

"port": 8020 //服务实例端口，这里要与Program类文件中WebHostBuilder.UseUrls()设置端口一致

}

}

}

1. **添加对配置文件appsettings.json的注册类ApplicationConfig.cs，如下：**

using Microsoft.Extensions.Configuration;//添加引用

using Steeltoe.Extensions.Configuration.CloudFoundry;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Web;

namespace AspNet461

{

public class ApplicationConfig

{

public static IConfigurationRoot Configuration { get; set; }

public static void RegisterConfig(string environment)

{

// Set up configuration sources.

var builder = new ConfigurationBuilder()

.SetBasePath(GetContentRoot())

.AddJsonFile("appsettings.json", optional: false, reloadOnChange: false)

.AddJsonFile($"appsettings.{environment}.json", optional: true)

.AddCloudFoundry()

.AddEnvironmentVariables();

Configuration = builder.Build();

}

public static string GetContentRoot()

{

var basePath = (string)AppDomain.CurrentDomain.GetData("APP\_CONTEXT\_BASE\_DIRECTORY") ??

AppDomain.CurrentDomain.BaseDirectory;

return Path.GetFullPath(basePath);

}

}

}

1. **在Global.cs的Application\_Start方法添加对Steeltoe的初始化，如下：**

//注册spring cloud配置信息

ApplicationConfig.RegisterConfig("development");

//autofac IoC容器

var builder = new ContainerBuilder();

// Add Microsoft Options to container

builder.RegisterOptions();

// Add Microsoft Logging to container

builder.RegisterLogging(ApplicationConfig.Configuration);

// Register your Web API controllers.

builder.RegisterApiControllers(Assembly.GetExecutingAssembly());

// Register IDiscoveryClient, etc.

builder.RegisterDiscoveryClient(ApplicationConfig.Configuration);

//设置全局依赖注入解析器

var container = builder.Build();

config.DependencyResolver = new AutofacWebApiDependencyResolver(container);

// Start the Discovery client background thread

container.StartDiscoveryClient();

1. **添加UserController.cs控制器，实现对其他接口服务(以User为例)的调用，如下：**

[RoutePrefix("api/User")]

public class UserController : ApiController

{

//请求消息处理器

private readonly DiscoveryHttpClientHandler \_handler;

//调用地址，这里指定应用名[user]，服务中心会自动分配地址，并实现负载均衡

private const string userUrl = "http://user/api/user";

/// <summary>

/// 构造器注入

/// </summary>

/// <param name="client"></param>

public UserController(IDiscoveryClient client)

{

\_handler = new DiscoveryHttpClientHandler(client);

}

/// <summary>

/// 调User服务中，获取用户信息

/// </summary>

/// <returns></returns>

[HttpGet]

public async Task<string> GetUserAsync()

{

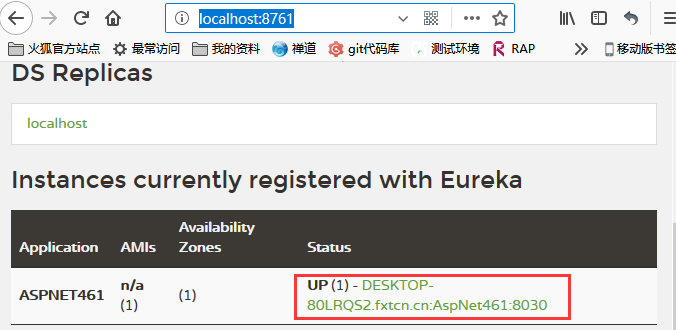
var client = new HttpClient(\_handler);

return await client.GetStringAsync(userUrl);

}

}

1. **运行项目，查看http://localhost:8761/，发现服务已注册到spring cloud服务中心：**



1. **项目运行后，访问UserController控制器的GetUserAsync方法，http://localhost:7586/api/user，看到如下图，说明调用成功：**

