Whats RestEase And How To Use RestEase On MicroService Architecture

Sumesh Sukumaran

Updated date Nov 02, 2020

7.2k 0 4

RestEaseDemo.zip

Introduction

Before I start explaining RestEase, let me give a brief about how we used to communicate between two APIs. We use HTTP protocol and we create an instance of HTTP Client for sending requests and receiving responses.

Here are a few drawbacks of using the HTTPClient:

- One call takes a lot of code lines, so you'll probably end up with an additional wrapper class (because you follow DRY, don't you?)
- Adding more support for features like Authorization headers, query parameters cause the wrapper class to be even bigger which pulls you away from the original task
- It's the developer's responsibility to dispose of the client.
- It's the developer's responsibility to deserialize the response.

But RestEase will make our life simpler. Basically, RestEase is a little type-safe REST API client library for .NET Framework 4.5 and higher and .NET Platform Standard 1.1, which aims to make interacting with remote REST endpoints easy, without adding unnecessary complexity. It won't work on platforms that don't support runtime code generation, including .NET Native and iOS.

Almost every aspect of RestEase can be overridden and customized, leading to a large level of flexibility.

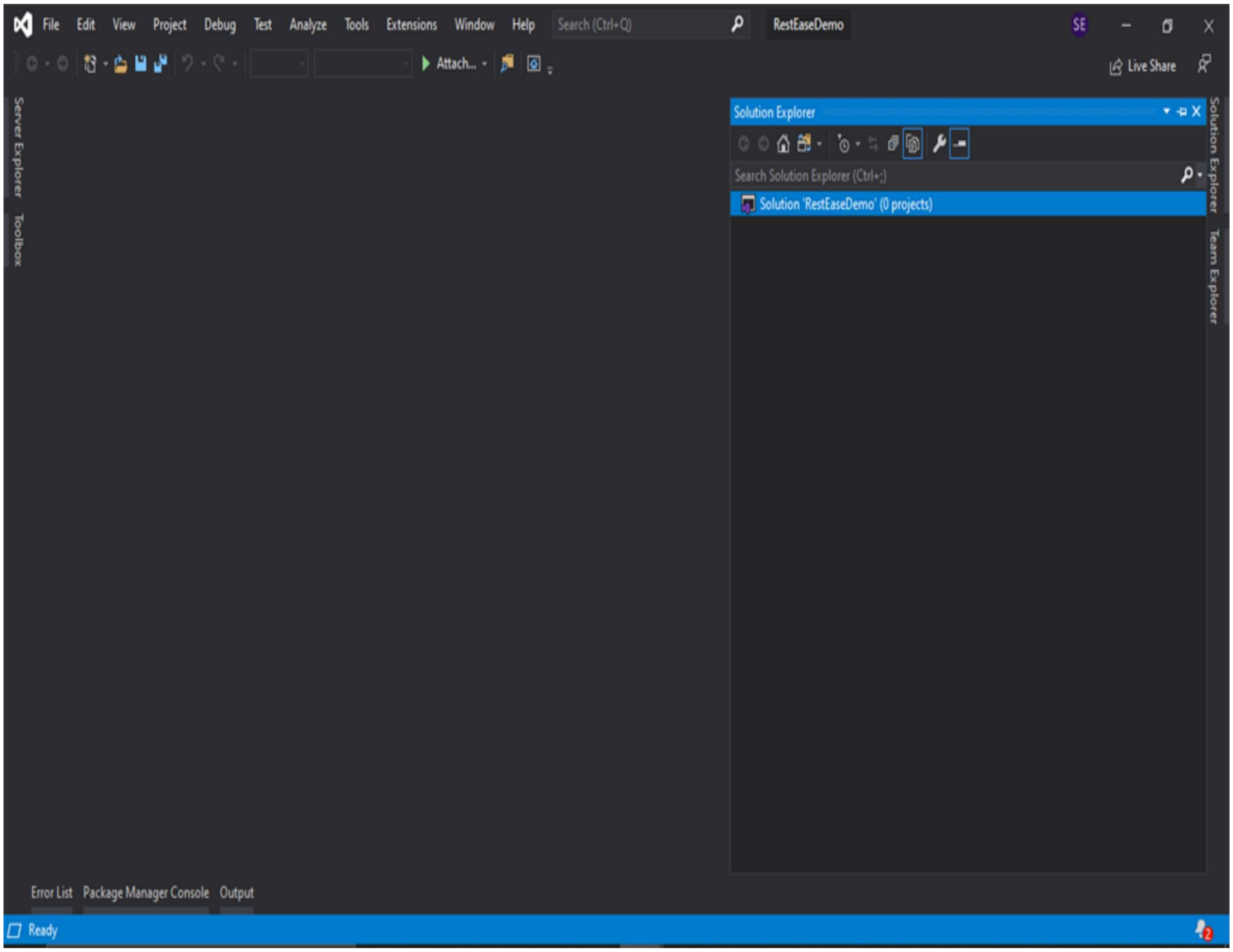
To use it, you define an interface(we can inject this service through the constructor which represents the endpoint you wish to communicate, where methods on that interface correspond to requests that can be made on it. RestEase will then generate an implementation of that interface for you, and by calling the methods you defined, the appropriate requests will be made.

RestEase is built on top of HttpClient and it is easy to gain access to the full capabilities of HttpClient, giving you control and flexibility, when you need it.

So let's see how we can implement RestEase in a microservice architecture.

Step 1

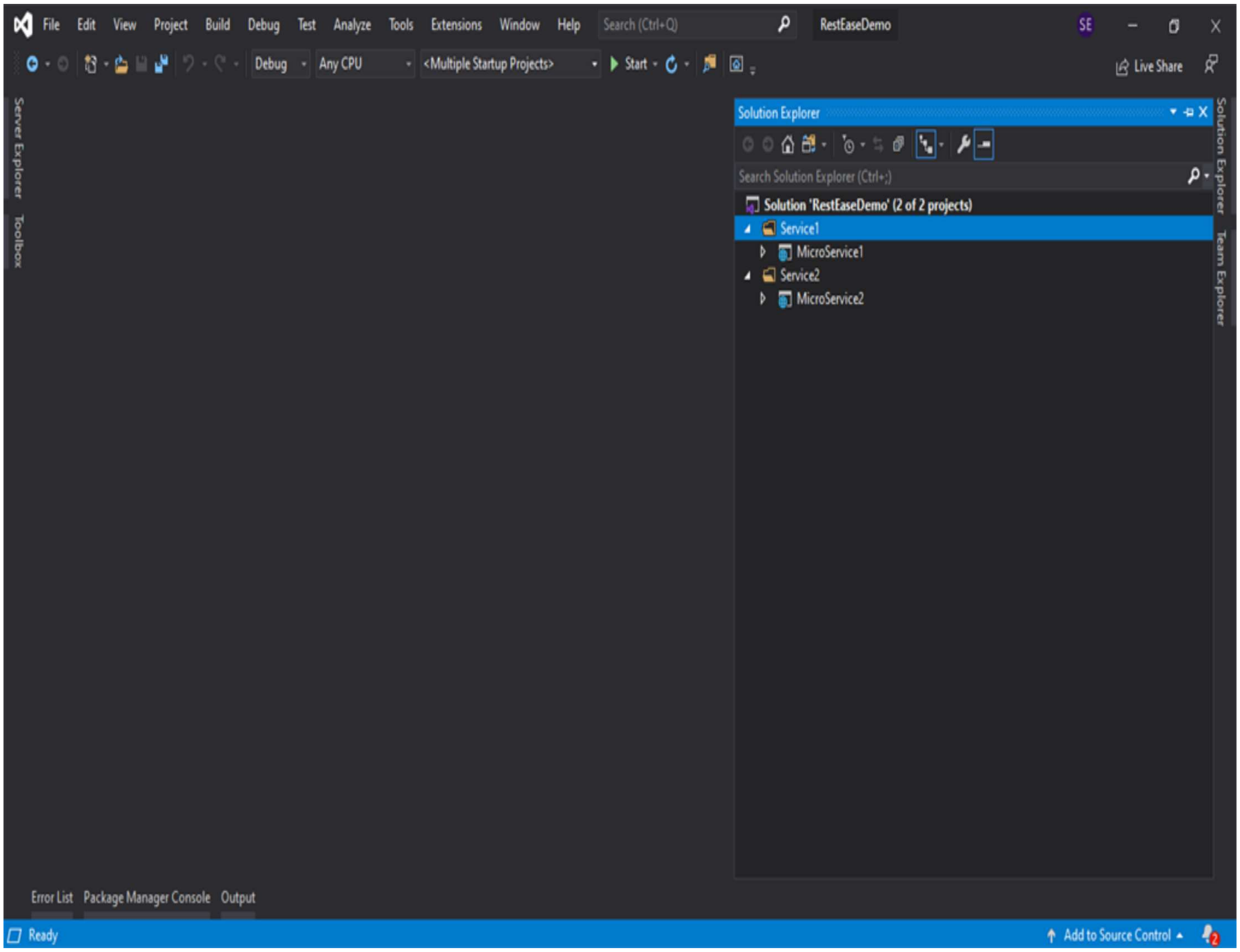
Add a blank solution:



Step 2

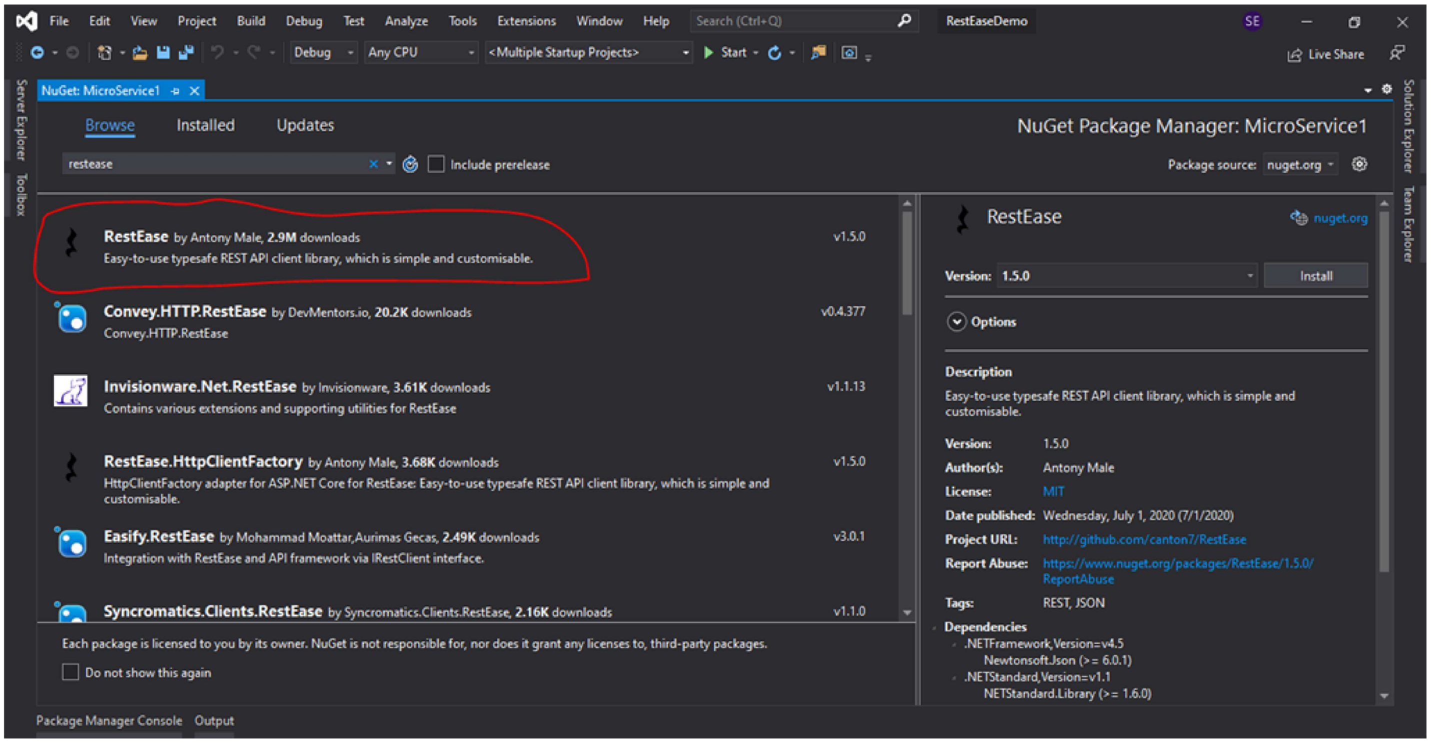
Create two folders under the Blank solutions, Service1 and Service2.

Add an API project, MicroService1, into Service1 and MicroService2 into Service2.



Step 3

Install RestEase on the MicroService1 project



NuGet Package Manager: MicroService1

restease Include prerelease Package source: nuget.org

RestEase by Antony Male, **2.9M** downloads v1.5.0
Easy-to-use typesafe REST API client library, which is simple and customisable.

Convey.HTTP.RestEase by DevMentors.io, **20.2K** downloads v0.4.377
Convey.HTTP.RestEase

Invisionware.Net.RestEase by Invisionware, **3.61K** downloads v1.1.13
Contains various extensions and supporting utilities for RestEase

RestEase.HttpClientFactory by Antony Male, **3.68K** downloads v1.5.0
HttpClientFactory adapter for ASP.NET Core for RestEase: Easy-to-use typesafe REST API client library, which is simple and customisable.

Easify.RestEase by Mohammad Moattar,Aurimas Gecas, **2.49K** downloads v3.0.1
Integration with RestEase and API framework via IRestClient interface.

Syncromatics.Clients.RestEase by Syncromatics.Clients.RestEase, **2.16K** downloads v1.1.0

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

Do not show this again

RestEase nuget.org

Version: 1.5.0 Install

Options

Description
Easy-to-use typesafe REST API client library, which is simple and customisable.

Version:	1.5.0
Author(s):	Antony Male
License:	MIT
Date published:	Wednesday, July 1, 2020 (7/1/2020)
Project URL:	http://github.com/canton7/RestEase
Report Abuse:	https://www.nuget.org/packages/RestEase/1.5.0/ReportAbuse
Tags:	REST, JSON

Dependencies
.NETFramework,Version=v4.5
Newtonsoft.Json (>= 6.0.1)
.NETStandard,Version=v1.1
NETStandard.Library (>= 1.6.0)

Once it's installed, add one interface, IPublishService.

```
[SerializationMethods(Query = QuerySerializationMethod.Serialized, Body =
public interface IPublishService {
    [AllowAnyStatusCode]
    [Get("Subscriber/getweather")]
    Task < IEnumerable < WeatherForecast >> GetWeather();
}
```

MicroService1 Controller will look like this:

```
[ApiController]
[Route("[controller]")]
public class PublishController: ControllerBase {
    private readonly IPublishService _publishService;
    public PublishController() {
            _publishService = RestClient.For < IPublishService > ("http:/,
        }
        [HttpGet]
    public async Task < IEnumerable < WeatherForecast >> Get() {
        var weather = await _publishService.GetWeather();
        return weather;
    }
}
```

MicroService2 Controller will look like this:

```
[ApiController]
[Route("[controller]")]
public class SubscriberController: ControllerBase {
    private static readonly string[] Summaries = new [] {
        "Freezing",
        "Bracing",
        "Chilly",
        "Cool",
        "Mild",
        "Warm",
        "Balmy",
        "Hot",
        "Sweltering",
        "Scorching"
    };
    private readonly ILogger < SubscriberController > _logger;
    public SubscriberController(ILogger < SubscriberController > logger) 
            _logger = logger;
        }
        [HttpGet]
        [Route("getweather")]
    public async Task < IEnumerable < WeatherForecast >> Get() {
        var rng = new Random();
        return Enumerable.Range(1, 5).Select(index => new WeatherForecast
            Date = DateTime.Now.AddDays(index),
                TemperatureC = rng.Next(-20, 55),
                Summary = Summaries[rng.Next(Summaries.Length)]
        }).ToArray();
    }
}
```

I have used ASP.NET Core 3.1 to create this project, and I am adding my whole project as a reference.

I hope this is helpful.

Please refer to this link for more information here.

ASP.net core   C#   HTTP Client   MicroService   Rest API   RestEase   Web API

Next Recommended Reading

Modern Architecture Shop (Clean Architecture And Microservices)

OUR BOOKS

Sumesh Sukumaran

1835 7.2k

4 0

Type your comment here and press Enter Key (Minimum 10 characters)

FEATURED ARTICLES

How To Upgrade to Windows 11

Exploring Subject <T> In Reactive Extensions For .Net

Micro Frontends With Webpack

What’s New In iPhone 13

Understanding Synchronization Context Task.ConfigureAwait In Action

View All

TRENDING UP

01 Getting Started With ASP.NET Web API .NET 5

02 Everything That Every .NET Developer Needs To Know About Disposable Types - Properly Disposing Objects

03 How To Upgrade ASP.NET Core 2.1 To ASP.NET Core 3.1 version

04 What Happens Behind The Scenes When You Trade A Bitcoin?

05 Developing A Web App Using Angular 12, ASP.NET Core Web API And SQL Server

06 Build A ToDo List Application Using Angular, .Net 5.0 Web API And Microsoft SQL Server

07 Need For Site Reliability Engineering

08 Micro Frontends With Webpack

09 Two Ways To Create Minimal APIs In .NET 6

10 Generate Image Thumbnail using Azure Computer Vision and Function App

View All