微软是推动WebDAV成为一个标准的主导力量,它自己利用自定义的HttpModule 实现

了IIs针对WebDAV的支持。但是这个默认注册(注册名称为 WebDAVModulc 会拒绝

HTTP方法为PuT和Delete的请求,如果我们的站点不需要提供针对WebDAV的支持,

解决这个问题最为直接的方式就是利用如下的配置将注册的HttpModu⒗ 移除。

<system.webServer>

<modules runAllManagedModulesForAllRequests="true">

<remove name="WebDAVModu1e" />

</modules>

<httpProtocol>

<customHeaders>

<add name="mydomain" value="www.sohu.com" />

<add name="Access-Control-Allow-Origin" value="\*" />

<add name="Access-Control-Allow-Methods" value="\*" />

<add name="Access-Control-Allow-Headers" value="Origin, X-Requested-With, Content-Type, Accept, Authorization"/>

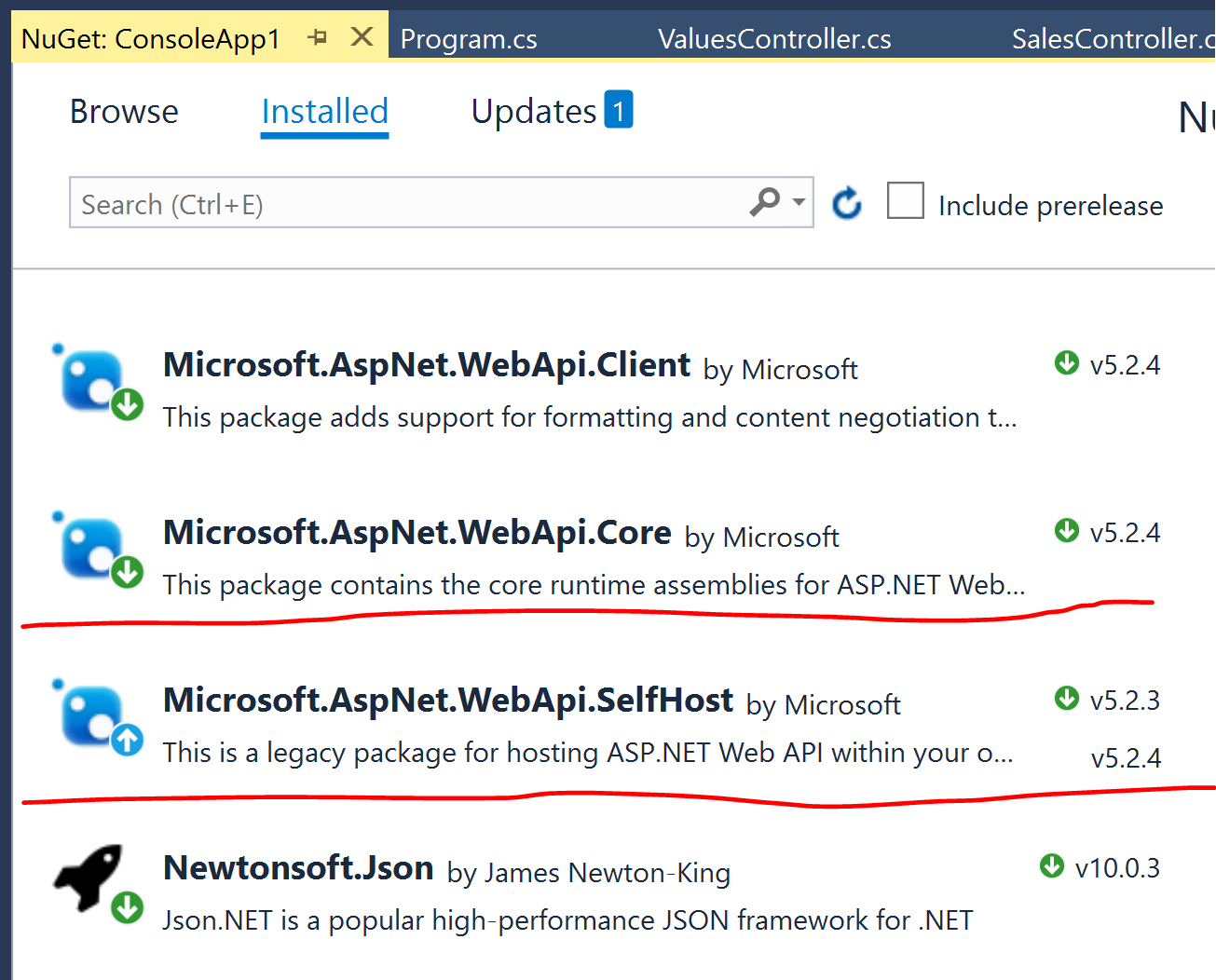
</customHeaders>

</httpProtocol>

</system.webServer>

让IIS 支持 Put , Delete , 同时让 Http Request 支持跨域访问。

Asp.Net WebApi 自定义寄存： 请Nuget Package: Microsoft.AspNet.WebApi.SelfHost



using System.Web.Http;

using System.Web.Http.SelfHost;

static void Main(string[] args)

{

HttpSelfHostConfiguration selfConfig = new HttpSelfHostConfiguration("http://localhost:3000/self");

selfConfig.Routes.MapHttpRoute(

name: "myApiRoute",

routeTemplate: "api/{controller}/{id}",

defaults: new { controller = "values", id = RouteParameter.Optional }

);

using (HttpSelfHostServer server = new HttpSelfHostServer(selfConfig))

{

server.OpenAsync().Wait();

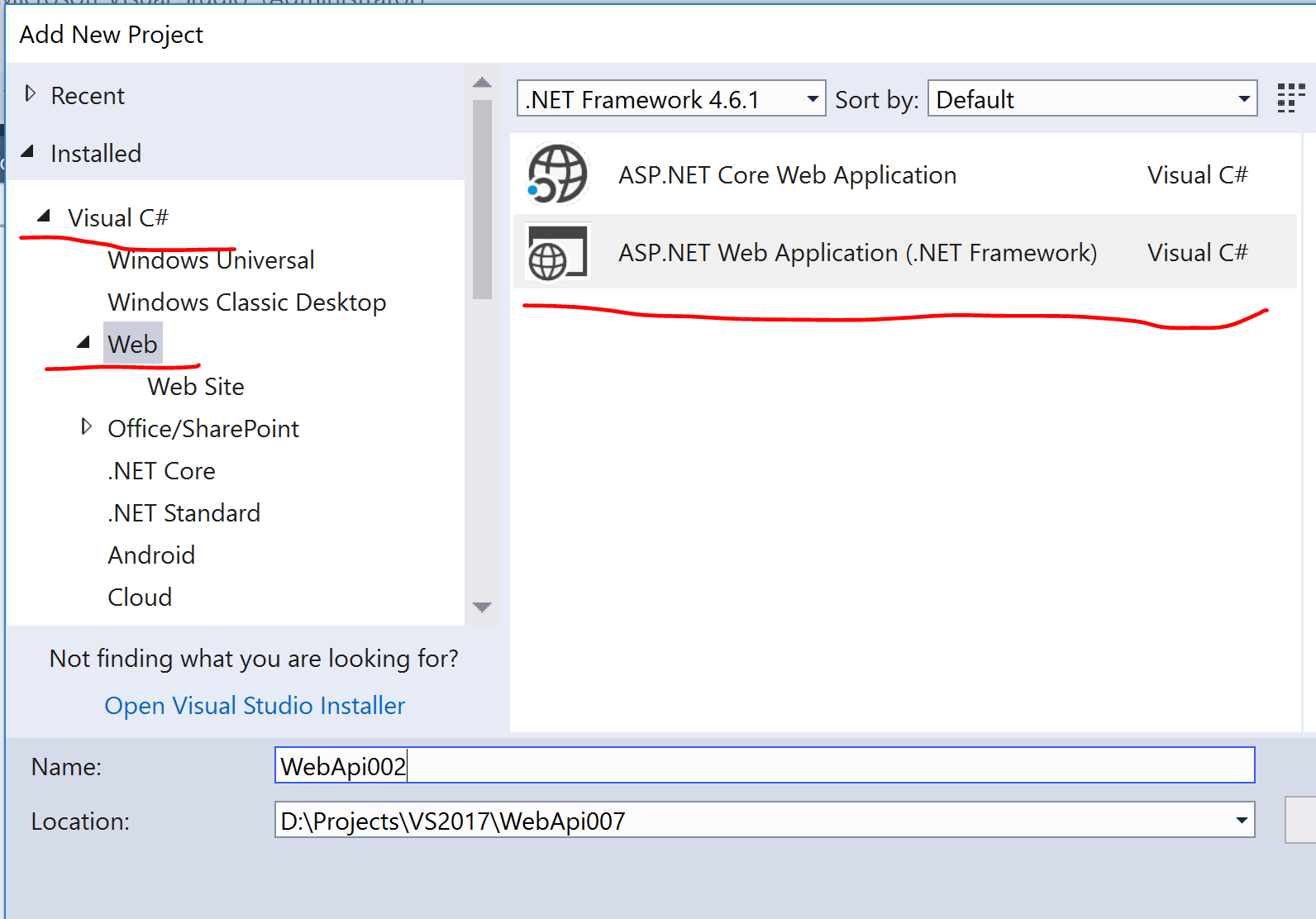
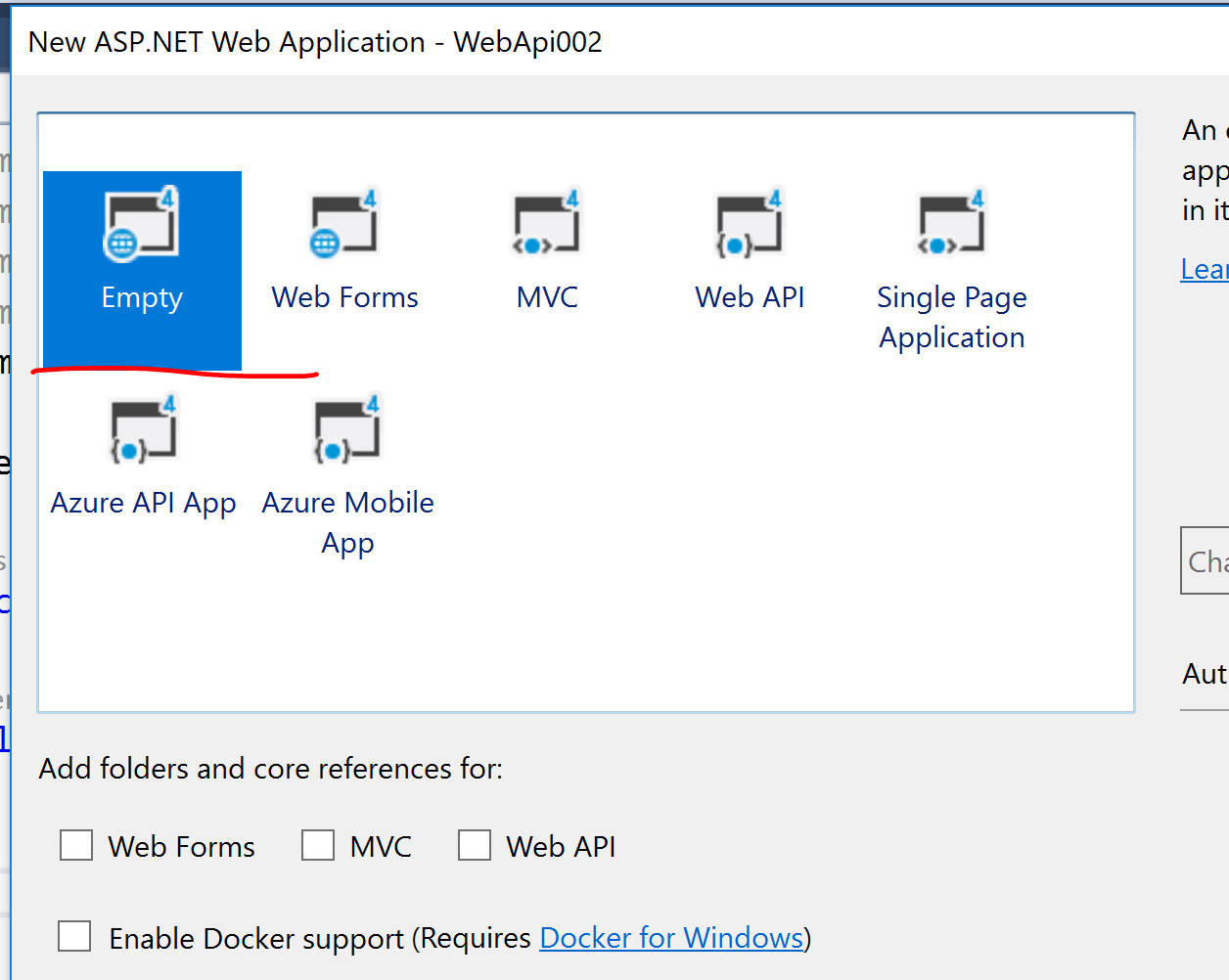
Console.WriteLine("My WebApi is started !");

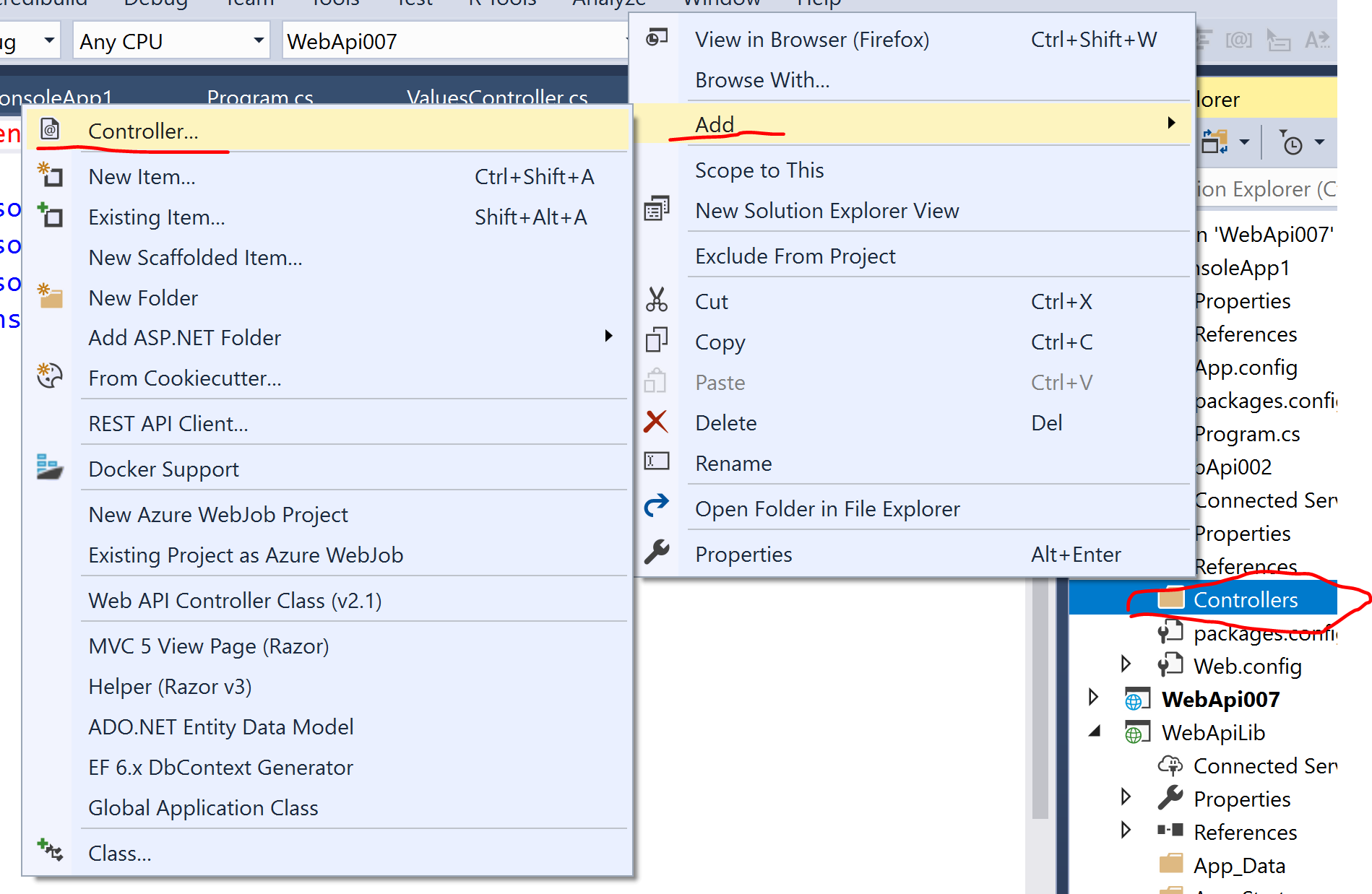
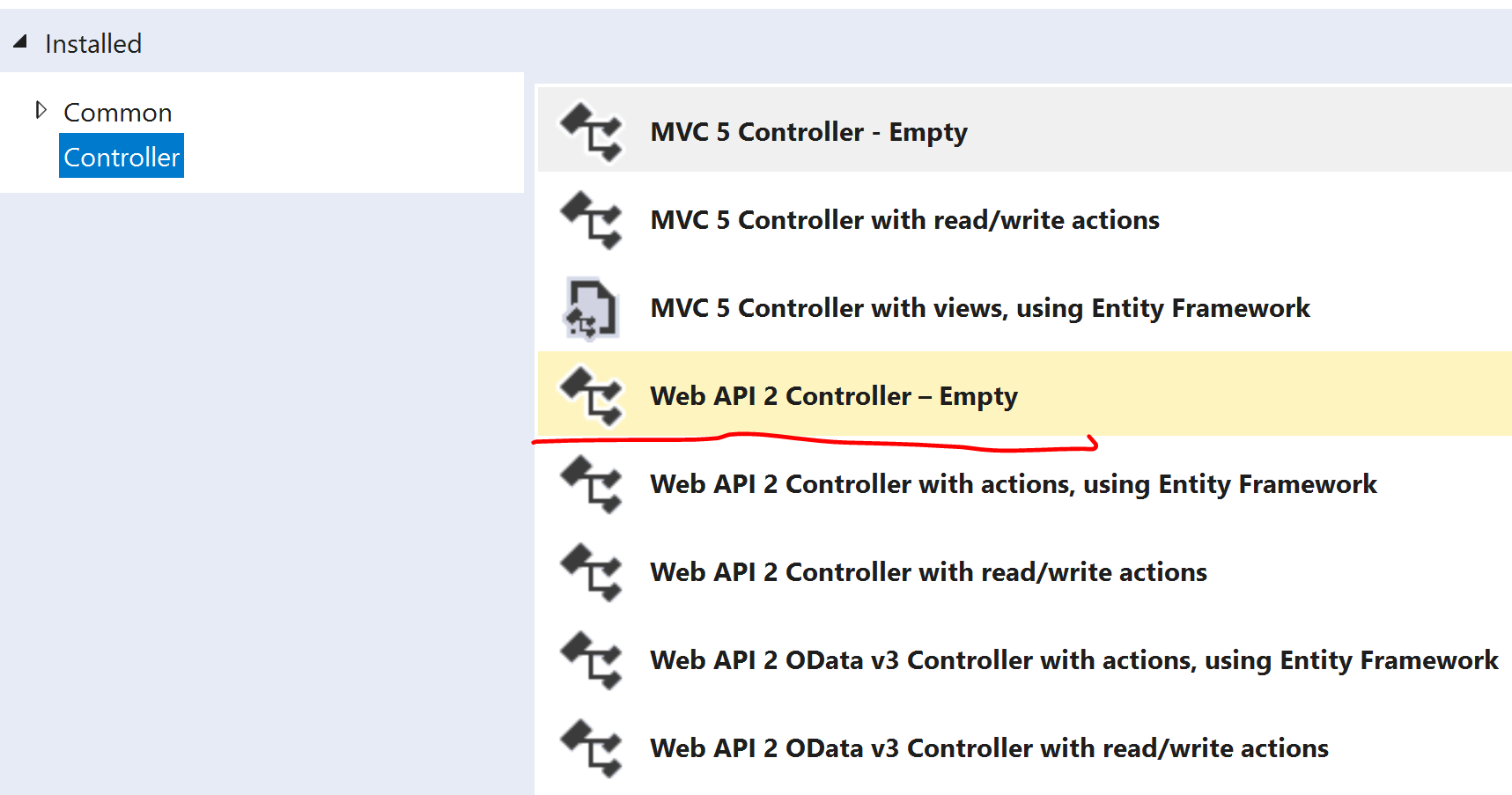
Console.ReadLine();

}

}

WebApi 定义： 可以创建一个空白的 Asp.Net

自定义寄宿SelfHost 引用WebApi 项目： WebApi002

注意只是引用 WebApi002 项目还是不够， 需要加载此程序集：

static void Main(string[] args)

{

Assembly.Load("WebApi002"); // important: load assembly

HttpSelfHostConfiguration selfConfig = new HttpSelfHostConfiguration("http://localhost:3000/self");

selfConfig.Routes.MapHttpRoute(

name: "myApiRoute",

routeTemplate: "api/{controller}/{id}",

defaults: new { controller = "sales", id = RouteParameter.Optional }

);

using (HttpSelfHostServer server = new HttpSelfHostServer(selfConfig))

{

server.OpenAsync().Wait();

Console.WriteLine("My WebApi is started !");

Console.ReadLine();

}

}

WebAPI 的Host 与 WebAPI 的业务逻辑Controllers 放在不同的项目里。 需要如此使用既可。

第一步： 先在Host 项目里引用 Reference WebAPI Controllers项目：

第二步： 需要使用代码加载程序集。

public static class WebApiConfig

{

public static void Register(HttpConfiguration config)

{

Assembly.Load("WebApiLib");

config.MapHttpAttributeRoutes();

config.Routes.MapHttpRoute(

name: "DefaultApi",

routeTemplate: "api/{controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

}

}

var pairs = new List<KeyValuePair<string, string>>

{

new KeyValuePair<string, string>("app\_key", "dKqgHG6c7xhNs2Kp"),

new KeyValuePair<string, string>("keywords", evt.navi.keywords),

new KeyValuePair<string, string>("location", evt.navi.location),

new KeyValuePair<string, string>("date", "Future"),

new KeyValuePair<string, string>("page\_number", evt.navi.pageNumber.ToString()),

new KeyValuePair<string, string>("page\_size", evt.navi.pageSize.ToString())

};

var content = new FormUrlEncodedContent(pairs);

HttpResponseMessage Content - public abstract class HttpContent : IDisposable

* FormUrlEncodedContent

var pairs = new List<KeyValuePair<string, string>>

{

new KeyValuePair<string, string>("app\_key", "dKqgHG6c7xhNs2Kp"),

new KeyValuePair<string, string>("keywords", evt.navi.keywords),

new KeyValuePair<string, string>("location", evt.navi.location),

new KeyValuePair<string, string>("date", "Future"),

new KeyValuePair<string, string>("page\_number", evt.navi.pageNumber.ToString()),

new KeyValuePair<string, string>("page\_size", evt.navi.pageSize.ToString())

};

var content = new FormUrlEncodedContent(pairs);

return: app\_key=dKqgHG6hNs2Kp&keywords=helloKey&location=Vancouver&date=Future&page\_number=558&page\_size=2018

* StringContent

StringBuilder sb = new StringBuilder();

sb.AppendLine("This is String Content");

sb.AppendLine("------------------------");

sb.AppendLine("Name, School, Address");

resp.Content = new StringContent(sb.ToString());

return:

This is String Content

------------------------

Name, School, Address

* ByteArrayContent

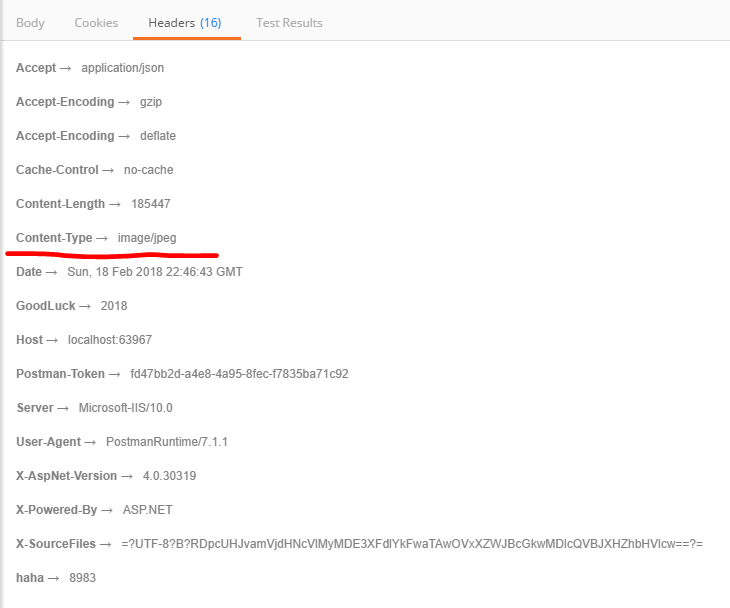
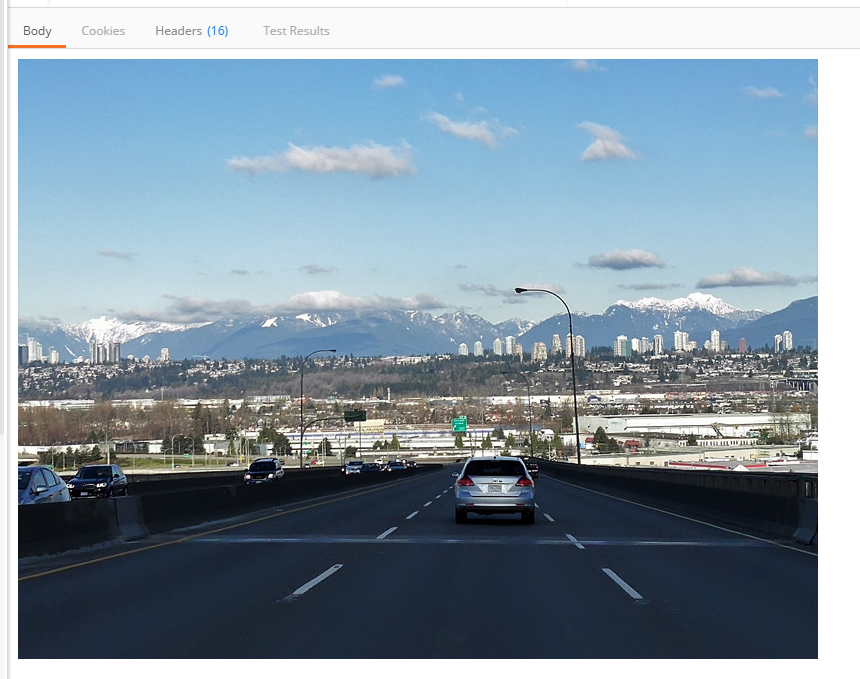
HttpResponseMessage resp = new HttpResponseMessage();

Byte[] buf = File.ReadAllBytes(@"d:\temp\lwh.jpg");

resp.Content = new ByteArrayContent(buf);

resp.Content.Headers.Add("content-type", "image/jpeg");

resp.Headers.Add("haha", "8983");

<img src="http://localhost:63967/API/values" /> 直接用于 html image

如果是想下载文件：

HttpResponseMessage resp = new HttpResponseMessage();

Byte[] buf = File.ReadAllBytes(@"d:\temp\lwh.jpg");

resp.Content = new ByteArrayContent(buf);

resp.Content.Headers.Add("content-type", "application/octet-stream");

resp.Content.Headers.Add("Content-Disposition", "Attachment; filename=lwh001.jpg");

记得头信息设置 Content-Type & Content-Disposition

* StreamContent

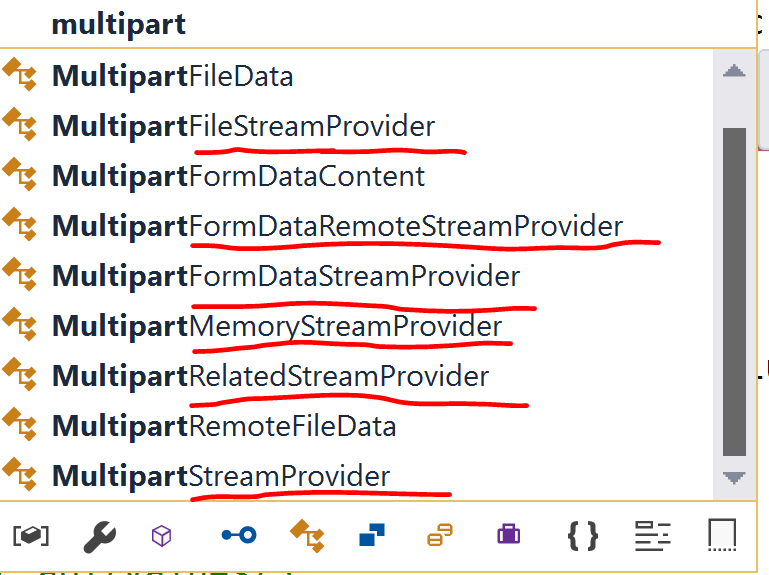
FileStream fs = File.OpenRead(@"d:\temp\lwh.jpg");

resp.Content = new StreamContent(fs);

resp.Content.Headers.Add("content-type", "application/octet-stream");

resp.Content.Headers.Add("Content-Disposition", "Attachment; filename=lwh001.jpg");

* MultipartContent



* 关于WebApi 接收 MultipartContent

如上图所示：有多个方法接收：

* MultipartMemoryStreamProvider

使用此方法的好处：

1. 接收到的内容是存放在内存里，可以通过 contents 来遍历每个文件，因为具体的文件内容在内存Stream里，因此可以灵活处理，如写入数据库，根据传递的头信息写入磁盘里。
2. 此方法可以接收所有的： ByteArrayContent， StreamContent，StringContent， FormUrlEncodedContent 等

public async Task<IHttpActionResult> Post()

{

if( Request.Content.IsMimeMultipartContent() )

{

// Read MultiPart Content to Provider: FileStream, MemoryStream etc

var str = await Request.Content.ReadAsStringAsync();

MultipartMemoryStreamProvider fsp = new MultipartMemoryStreamProvider();

await Request.Content.ReadAsMultipartAsync(fsp);

var i = 1;

foreach(var file in fsp.Contents)

{

var fileType = file.Headers.GetValues("FileType").FirstOrDefault()??"unknowType";

var fileName = file.Headers.GetValues("UploadFileName").FirstOrDefault()??"unknowFile";

FileStream fs = File.OpenWrite($"d:\\temp1\\{fileType}\_{fileName}");

await file.CopyToAsync(fs);

fs.Flush();

fs.Close();

//var t1 = file.LocalFileName;

//var t2 = file.Headers;

i++;

}

}

return Content(HttpStatusCode.OK, "Post Method Done");

}

-----Client Post-------------------------------------

HttpClient htp = new HttpClient();

MultipartContent contents = new MultipartContent();

FileStream fs = File.OpenRead(@"d:\temp\lwh.jpg");

StreamContent scontent = new StreamContent(fs);

scontent.Headers.ContentDisposition = new System.Net.Http.Headers.ContentDispositionHeaderValue("attachment") { FileName = "lwh0001.jpg", Name = "lwhname.jpg" };

scontent.Headers.ContentType = new System.Net.Http.Headers.MediaTypeHeaderValue("application/octet-stream");

scontent.Headers.Add("UploadFileName", "MyUpload\_01.jpg");

scontent.Headers.Add("FileType", "image\_jpg");

contents.Add(scontent);

FileStream fs1 = File.OpenRead(@"d:\temp\Interview.docx");

StreamContent dcontent = new StreamContent(fs1);

dcontent.Headers.ContentDisposition = new System.Net.Http.Headers.ContentDispositionHeaderValue("attachment") { FileName = "lwh0001.jpg", Name = "lwhname.jpg" };

dcontent.Headers.ContentType = new System.Net.Http.Headers.MediaTypeHeaderValue("application/octet-stream");

dcontent.Headers.Add("UploadFileName", "MyDocument\_01.doc");

dcontent.Headers.Add("FileType", "docu\_type");

contents.Add(dcontent);

Byte[] buf = File.ReadAllBytes(@"d:\temp\lwh1.jpg");

ByteArrayContent bcontent = new ByteArrayContent(buf);

bcontent.Headers.ContentDisposition = new System.Net.Http.Headers.ContentDispositionHeaderValue("attachment") { FileName = "lwh0001.jpg", Name="lwhname.jpg" };

bcontent.Headers.ContentType = new System.Net.Http.Headers.MediaTypeHeaderValue("application/octet-stream");

bcontent.Headers.Add("UploadFileName", "MyUpload\_02.jpg");

bcontent.Headers.Add("FileType", "image\_png");

contents.Add(bcontent);

var pairs = new List<KeyValuePair<string, string>>

{

new KeyValuePair<string, string>("app\_key", "dKqgHG6c7xhNs2Kp"),

new KeyValuePair<string, string>("date", "Future"),

new KeyValuePair<string, string>("page\_number", "500"),

new KeyValuePair<string, string>("page\_size", "25")

};

var content = new FormUrlEncodedContent(pairs);

content.Headers.Add("UploadFileName", "keyValue.txt");

content.Headers.Add("FileType", "keyValType");

contents.Add(content);

StringContent strcontent = new StringContent("Hello world");

strcontent.Headers.Add("UploadFileName", "MyString.txt");

strcontent.Headers.Add("FileType", "text\_asc");

contents.Add(strcontent);

HttpResponseMessage resp = htp.PostAsync("http://localhost:63967/API/values", contents).Result;

if( resp.IsSuccessStatusCode )

{

Console.WriteLine(resp.Content.ReadAsStringAsync().Result);

} else

{

Console.WriteLine(resp.Content.ReadAsStringAsync().Result);

}

我们可以不必自定义头信息来记录文件的信息：可以使用已有的属性来定义文件信息：

public async Task<IHttpActionResult> Post()

{

try

{

if (Request.Content.IsMimeMultipartContent())

{

// Read MultiPart Content to Provider: FileStream, MemoryStream etc

var str = await Request.Content.ReadAsStringAsync();

MultipartMemoryStreamProvider fsp = new MultipartMemoryStreamProvider();

await Request.Content.ReadAsMultipartAsync(fsp);

var i = 0;

foreach (var file in fsp.Contents)

{

i++;

var fileName = file.Headers?.ContentDisposition?.FileName??"unknowFname";

var name = file.Headers?.ContentDisposition?.Name??"UnkownName";

var fileType = file.Headers?.ContentType?.MediaType??"UnknowType";

FileStream fs = File.OpenWrite($"d:\\temp1\\{name}\_{fileName}");

await file.CopyToAsync(fs);

fs.Flush();

fs.Close();

}

}

return Content(HttpStatusCode.OK, "Post Method Done");

}

catch (Exception err)

{

return Content(HttpStatusCode.BadRequest, $"{err.Message}\n\n{err.StackTrace}");

}

}

--------Client Post -----------------------

HttpClient htp = new HttpClient();

MultipartContent contents = new MultipartContent();

FileStream fs = File.OpenRead(@"d:\temp\lwh.jpg");

StreamContent scontent = new StreamContent(fs);

scontent.Headers.ContentDisposition = new System.Net.Http.Headers.ContentDispositionHeaderValue("attachment") { FileName = "lwh0001.jpg", Name = "LWH01" };

scontent.Headers.ContentType = new System.Net.Http.Headers.MediaTypeHeaderValue("application/octet-stream");

contents.Add(scontent);

FileStream fs1 = File.OpenRead(@"d:\temp\Interview.docx");

StreamContent dcontent = new StreamContent(fs1);

dcontent.Headers.ContentDisposition = new System.Net.Http.Headers.ContentDispositionHeaderValue("attachment") { FileName = "MyDocu.docx", Name = "DOC01" };

dcontent.Headers.ContentType = new System.Net.Http.Headers.MediaTypeHeaderValue("application/octet-stream");

contents.Add(dcontent);

Byte[] buf = File.ReadAllBytes(@"d:\temp\lwh1.jpg");

ByteArrayContent bcontent = new ByteArrayContent(buf);

bcontent.Headers.ContentDisposition = new System.Net.Http.Headers.ContentDispositionHeaderValue("attachment") { FileName = "lwh0002.jpg", Name="LWH02" };

bcontent.Headers.ContentType = new System.Net.Http.Headers.MediaTypeHeaderValue("application/octet-stream");

contents.Add(bcontent);

var pairs = new List<KeyValuePair<string, string>>

{

new KeyValuePair<string, string>("app\_key", "dKqgHG6c7xhNs2Kp"),

new KeyValuePair<string, string>("date", "Future"),

new KeyValuePair<string, string>("page\_number", "500"),

new KeyValuePair<string, string>("page\_size", "25")

};

var content = new FormUrlEncodedContent(pairs);

content.Headers.ContentDisposition = new System.Net.Http.Headers.ContentDispositionHeaderValue("attachment") { FileName = "lwtText01.txt", Name = "TXT01" };

content.Headers.ContentType = new System.Net.Http.Headers.MediaTypeHeaderValue("application/octet-stream");

contents.Add(content);

StringContent strcontent = new StringContent("Hello world");

strcontent.Headers.ContentDisposition = new System.Net.Http.Headers.ContentDispositionHeaderValue("attachment") { FileName = "lwtText02.txt", Name = "TXT02" };

strcontent.Headers.ContentType = new System.Net.Http.Headers.MediaTypeHeaderValue("application/octet-stream");

contents.Add(strcontent);

HttpResponseMessage resp = htp.PostAsync("http://localhost:63967/API/values", contents).Result;

if( resp.IsSuccessStatusCode )

{

Console.WriteLine(resp.Content.ReadAsStringAsync().Result);

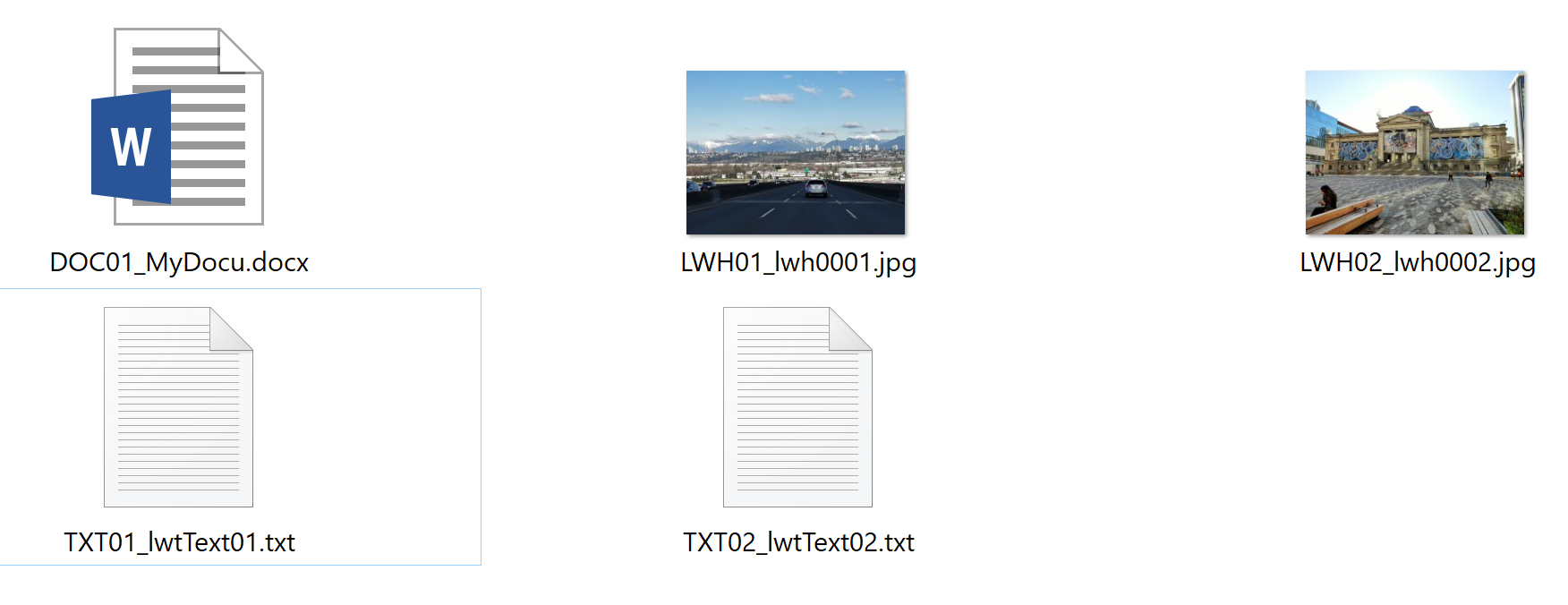
} else

{

Console.WriteLine(resp.Content.ReadAsStringAsync().Result);

}

------ Result --------



* MultipartFileStreamProvider

注意使用此方法：

1. 一下代码执行完后，文件被写入到指定的目录里了，是系统生成的临时文件，而且没有后缀名

MultipartFileStreamProvider fsp = new MultipartFileStreamProvider(@"d:\temp1");

await Request.Content.ReadAsMultipartAsync(fsp);

1. 如果我们还想使用stream 将文件单独再使用stream写入到存储介质。将会出错。

因为文件流已经被读取并清空了。

foreach (var file in fsp.Contents)

{

i++;

var fileName = file.Headers?.ContentDisposition?.FileName??"unknowFname";

var name = file.Headers?.ContentDisposition?.Name??"UnkownName";

var fileType = file.Headers?.ContentType?.MediaType??"UnknowType";

FileStream fs = File.OpenWrite($"d:\\temp1\\{name}\_{fileName}");

await file.CopyToAsync(fs);

fs.Flush();

fs.Close();

}

1. 此方法，只能将临时文件根据头信息提供的文件信息重新命名成自己想要的名字

获取临时文件的信息只能透过 fsp.FileData

foreach (var file in fsp.FileData)

{

i++;

var sourceFileName = file.LocalFileName;

var sourceFile = Path.Combine(@"d:\temp1", sourceFileName);

var fileName = file.Headers?.ContentDisposition?.FileName??"unknowFname";

var name = file.Headers?.ContentDisposition?.Name??"UnkownName";

var fileType = file.Headers?.ContentType?.MediaType??"UnknowType";

var destFile = Path.Combine(@"d:\temp1", $"{name}\_{fileName}");

File.Move(sourceFile, destFile);

}

---- 完整的WebAPI -----

public async Task<IHttpActionResult> Post()

{

try

{

if (Request.Content.IsMimeMultipartContent())

{

// Read MultiPart Content to Provider: FileStream, MemoryStream etc

var str = await Request.Content.ReadAsStringAsync();

MultipartFileStreamProvider fsp = new MultipartFileStreamProvider(@"d:\temp1");

await Request.Content.ReadAsMultipartAsync(fsp);

var i = 0;

foreach (var file in fsp.FileData)

{

i++;

var sourceFileName = file.LocalFileName;

var sourceFile = Path.Combine(@"d:\temp1", sourceFileName);

var fileName = file.Headers?.ContentDisposition?.FileName??"unknowFname";

var name = file.Headers?.ContentDisposition?.Name??"UnkownName";

var fileType = file.Headers?.ContentType?.MediaType??"UnknowType";

var destFile = Path.Combine(@"d:\temp1", $"{name}\_{fileName}");

File.Move(sourceFile, destFile);

}

}

return Content(HttpStatusCode.OK, "Post Method Done");

}

catch (Exception err)

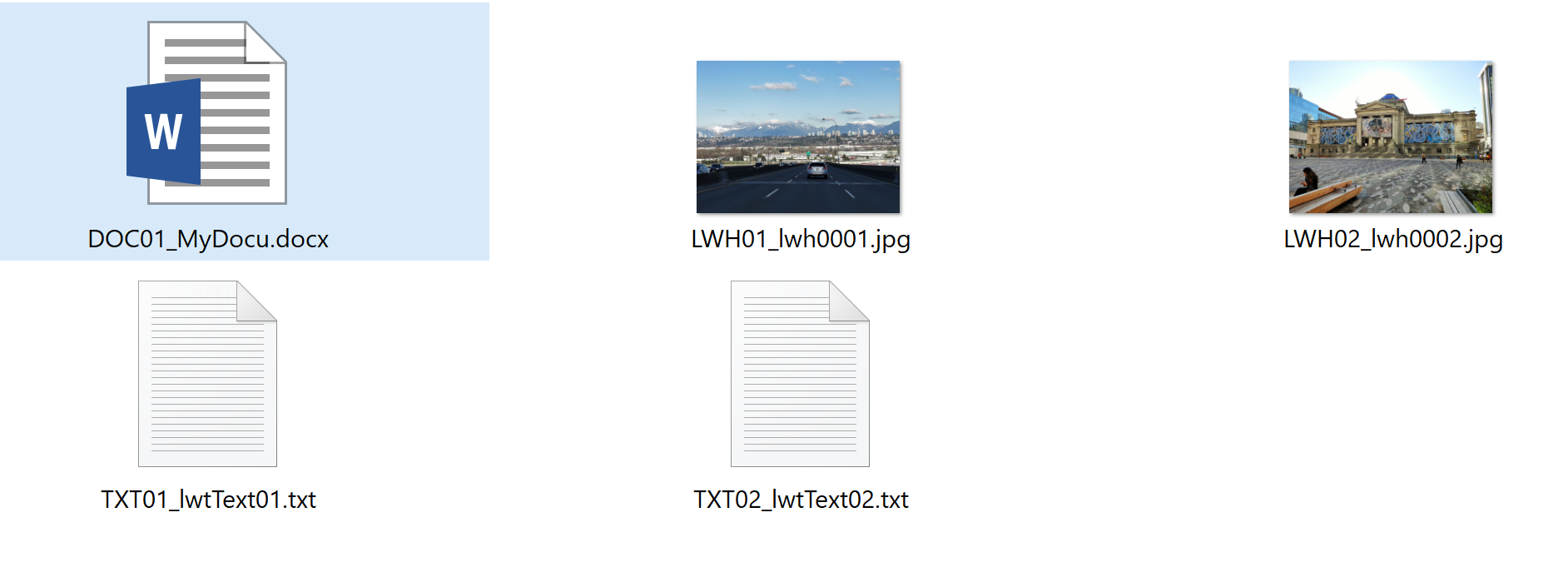
{

return Content(HttpStatusCode.BadRequest, $"{err.Message}\n\n{err.StackTrace}");

}

}

--- 结果如下 -----



使用HTML Post - multipart/form-data 实现同时上传多个文件：

<form action="http://localhost:63967/API/values?idd=988&myname=William"

enctype="multipart/form-data" method="post">

<input type="text" name="firstName" value="Lilian" />

<input type="text" name="lastName" value="Liu" />

<input type="file" name="file1" value="" />

<input type="file" name="file2" value="" />

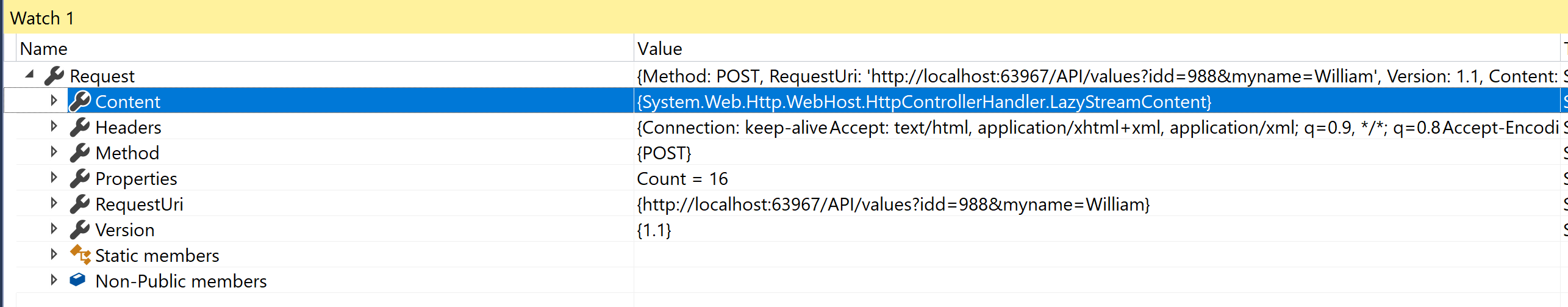
<input type="file" name="file3" value="" />

<input type="submit" name="submit" value="submit" />

</form>

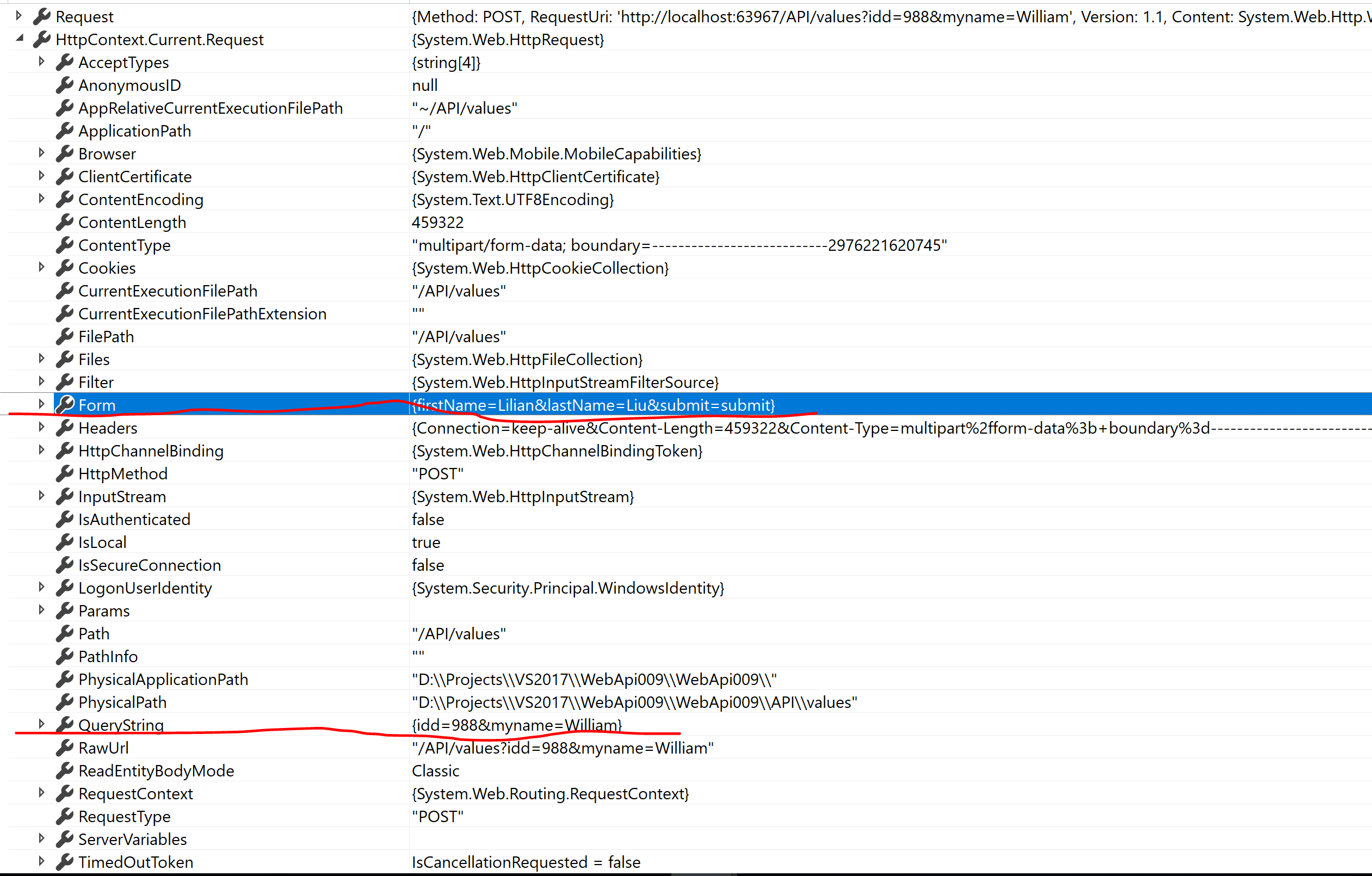
特别注意： ApiController 里的 Request 对象 和 MVC 里的Request对象是不同的对象。理解这一点非常重要：

ApiController : Request 对象：



在WebApi ApiController 要获取 HttpContext.Current.Request 对象

using System.Web;



public async Task<IHttpActionResult> Post(string firstName, string lastName, string idd="", string myname="" )

{

try

{

var itt = idd;

var nnn = myname;

var fn = firstName;

var ln = lastName;

if (Request.Content.IsMimeMultipartContent())

{

// Read MultiPart Content to Provider: FileStream, MemoryStream etc

MultipartFileStreamProvider fsp = new MultipartFileStreamProvider(@"d:\temp1");

await Request.Content.ReadAsMultipartAsync(fsp);

var i = 0;

foreach (var file in fsp.FileData)

{

i++;

var sourceFileName = file.LocalFileName;

var sourceFile = Path.Combine(@"d:\temp1", sourceFileName);

var fileName = file.Headers?.ContentDisposition?.FileName ?? $"params\_{i}.txt";

fileName = fileName.Replace("\"", "");

fileName = fileName.Replace("\\", "");

var name = file.Headers?.ContentDisposition?.Name ?? "UnkownName";

var fileType = file.Headers?.ContentType?.MediaType ?? "UnknowType";

var destFile = Path.Combine(@"d:\temp1", $"{fileName}");

File.Move(sourceFile, destFile);

}

}

return Content(HttpStatusCode.OK, "Post Method Done");

}

catch (Exception err)

{

return Content(HttpStatusCode.BadRequest, $"{err.Message}\n\n{err.StackTrace}");

}

}

如果我们提交，将会出错： 原因在于

string firstName, string lastName 并不在自动绑定数据模型里：

因为是使用ApiController Post 他们是在multipart/form-data 里的一部分

var fn = HttpContext.Current.Request.Form["firstName"];

var ln = HttpContext.Current.Request.Form["lastName"];

但是string idd="", string myname="" , QueryString 里带的数据可以被自动绑定到。

-----修正之后代码如下，工作正常 --------

public async Task<IHttpActionResult> Post(string idd="", string myname="" )

{

try

{

var itt = idd;

var nnn = myname;

//可以如此来访问 Post 过来的 FormData

var fn = HttpContext.Current.Request.Form["firstName"];

var ln = HttpContext.Current.Request.Form["lastName"];

if (Request.Content.IsMimeMultipartContent())

{

// Read MultiPart Content to Provider: FileStream, MemoryStream etc

MultipartFileStreamProvider fsp = new MultipartFileStreamProvider(@"d:\temp1");

await Request.Content.ReadAsMultipartAsync(fsp);

var i = 0;

foreach (var file in fsp.FileData)

{

i++;

var sourceFileName = file.LocalFileName;

var sourceFile = Path.Combine(@"d:\temp1", sourceFileName);

var fileName = file.Headers?.ContentDisposition?.FileName ?? $"params\_{i}.txt";

fileName = fileName.Replace("\"", "");

fileName = fileName.Replace("\\", "");

var name = file.Headers?.ContentDisposition?.Name ?? "UnkownName";

var fileType = file.Headers?.ContentType?.MediaType ?? "UnknowType";

var destFile = Path.Combine(@"d:\temp1", $"{fileName}");

File.Move(sourceFile, destFile);

}

}

return Content(HttpStatusCode.OK, "Post Method Done");

}

catch (Exception err)

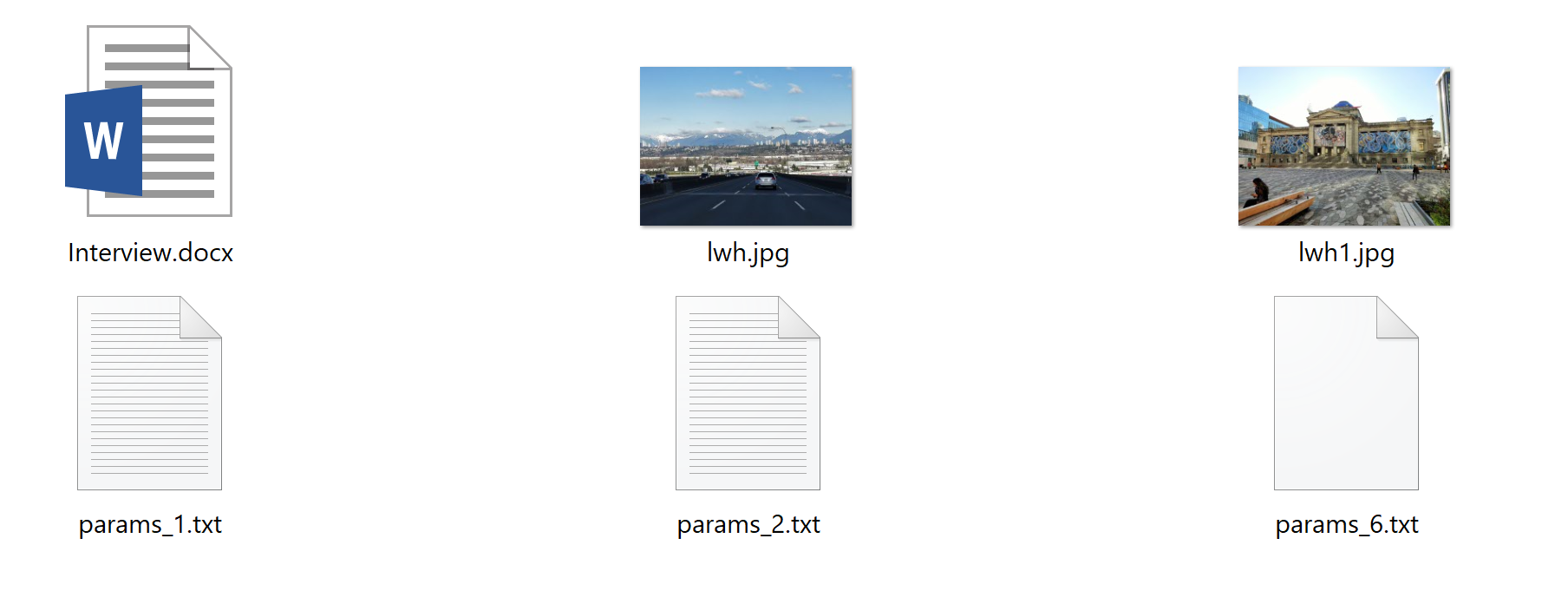
{

return Content(HttpStatusCode.BadRequest, $"{err.Message}\n\n{err.StackTrace}");

}

}

输出结果：



params\_1.txt : 内容： Lilian

params\_2.txt : 内容： Liu

params\_6.txt : 内容： submit submit 按钮的值

此方案缺点是： 代码复杂，而且把所有的FormData 都当成单独的文件来处理。

----可以使用数组来传递多个文件------

<form action="http://localhost:63967/API/values?idd=988&myname=William" enctype="multipart/form-data" method="post">

<input type="text" name="firstName" value="Lilian" />

<input type="text" name="lastName" value="Liu" />

<input type="file" name="file[]" value="" />

<input type="file" name="file[]" value="" />

<input type="file" name="file[]" value="" />

<input type="submit" name="submit" value="submit" />

</form>

--------还有最简单的解决方案：使用 **HttpContext.Current.Request.Files** --------------------------

public async Task<IHttpActionResult> Post(string idd="", string myname="")

{

try

{

var itt = idd;

var nnn = myname;

var fn = HttpContext.Current.Request.Form["firstName"];

var ln = HttpContext.Current.Request.Form["lastName"];

var files = HttpContext.Current.Request.Files;

foreach(string fileKey in files.AllKeys)

{

string destFile = Path.Combine(@"d:\temp1", files[fileKey].FileName);

files[fileKey].SaveAs(destFile);

}

return Content(HttpStatusCode.OK, "Post Method Done");

}

catch (Exception err)

{

return Content(HttpStatusCode.BadRequest, $"{err.Message}\n\n{err.StackTrace}");

}

}

-----如果不传文件 ： application/x-www-form-urlencoded ------

<form action="http://localhost:63967/API/values?idd=988&myname=William"

enctype="application/x-www-form-urlencoded" method="post">

<input type="text" name="firstName" value="Lilian" />

<input type="text" name="lastName" value="Liu" />

<input type="submit" name="submit" value="submit" />

</form>

--- webapi: ----------------

if (Request.Content.IsFormData())

{

NameValueCollection kvs = await Request.Content.ReadAsFormDataAsync();

}

案例： file 数组

<form action="http://localhost:63967/API/values?idd=988&myname=William" enctype="multipart/form-data" method="post">

<input type="text" name="firstName" value="Lilian" />

<input type="text" name="lastName" value="Liu" />

<input type="file" name="file" value="" />

<input type="file" name="file" value="" />

<input type="file" name="file" value="" />

<input type="submit" name="submit" value="submit" />

</form>

<form action="http://localhost:63967/API/values?idd=988&myname=William" enctype="multipart/form-data" method="post">

<input type="text" name="firstName" value="Lilian" />

<input type="text" name="lastName" value="Liu" />

<input type="file" name="file[]" value="" />

<input type="file" name="file[]" value="" />

<input type="file" name="file[]" value="" />

<input type="submit" name="submit" value="submit" />

</form>s

--- 无论使用：file 或者 file[] ; 传上去的文件数是 3 个， 可只保存一个文件 ----

var files = HttpContext.Current.Request.Files;

foreach (string fileKey in files.AllKeys)

{

string destFile = Path.Combine(@"d:\temp1", files[fileKey].FileName);

files[fileKey].SaveAs(destFile);

}

----- 修改以后， 工作正常保存3个 -----

var files = HttpContext.Current.Request.Files;

for (int fileKey=0; fileKey<files.AllKeys.Length; fileKey++ )

{

string destFile = Path.Combine(@"d:\temp1", files[fileKey].FileName);

files[fileKey].SaveAs(destFile);

}

使用这个代码也是没有问题的：

if (Request.Content.IsMimeMultipartContent())

{

MultipartFileStreamProvider fsp = new MultipartFileStreamProvider(@"d:\temp1");

await Request.Content.ReadAsMultipartAsync(fsp);

var i = 0;

foreach (var file in fsp.FileData)

{

i++;

var sourceFileName = file.LocalFileName;

var sourceFile = Path.Combine(@"d:\temp1", sourceFileName);

var fileName = file.Headers?.ContentDisposition?.FileName ?? $"params\_{i}.txt";

fileName = fileName.Replace("\"", "");

fileName = fileName.Replace("\\", "");

var name = file.Headers?.ContentDisposition?.Name ?? "UnkownName";

var fileType = file.Headers?.ContentType?.MediaType ?? "UnknowType";

var destFile = Path.Combine(@"d:\temp1", $"{fileName}");

File.Move(sourceFile, destFile);

}

}