今天在研究WebAPI的上传与下载，作为Rest的框架，更多是面向资源，就其本身来说，是不会涉及也不应该涉及到大文件的处理，具体多大呢，也就是ASP.NET的限值2G。
ASP.NET的pipeline对于上传文件一般的处理流程是接收到文件，放到内存中，我们也一般只处理后续的流程，例如存入硬盘等等。
目前想象的一个场景是上传一个大文件，后续处理可能有多种。如果并发数过多，肯定会造成内存溢出，所以参考资料与琢磨，搞定了这个问题。

但是场景本身是有一定问题的，这纯属个人爱好在玩，真实场景来说文件不宜过大，不是专门处理大文件的服务器与协议，无非就是用流的方式，但是大文件传输使用流采用可靠协议是最好的，可靠不是指TCP的可靠，而是指在如果断开后，后续的处理，例如断点。
如果是HTTP的断点，那就只有自己写了。

一、配置文件修改

不多说，这个必须的。

```
1   <system.web>
2     <compilation debug="true" targetFramework="4.5" />
3     <httpRuntime targetFramework="4.5" maxRequestLength="2147483647"/>
4   </system.web>

    <security>
      <requestFiltering >
        <requestLimits maxAllowedContentLength="2147483647" ></requestLimits>
      </requestFiltering>
    </security>
```

二、扩展主机缓存Policy

这是微软预留的接口，作为扩展，让用户自己关闭主机是否缓冲，也就是正常流程中将上传文件的内容存入内存的动作

```
public class NoBufferPolicySelector : WebHostBufferPolicySelector
{
   public override bool UseBufferedInputStream(object hostContext)
   {
      var context = hostContext as HttpContextBase;

      if (context != null)
      {
          if (context.Request.HttpMethod == HttpMethod.Post.ToString() &&
context.Request.ContentLength >200000)
```

```
            return false;
        }

        return true;
    }
```

还有个动作就是要去注册它

```
        public static void Register(HttpConfiguration config)
        {
            // Web API 配置和服务

            // Web API 路由
            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );


GlobalConfiguration.Configuration.Services.Replace(typeof(IHostBufferPolicySelector),
new NoBufferPolicySelector());


        }
```

三、想怎么玩儿怎么玩儿

上传的代码，大家都懂的

```
 [RoutePrefix("test")]
    public class UploadController : ApiController
    {
        [Route("upload")]
        [HttpPost]
        public Task<IEnumerable<FileDesc>> Post()
        {
```

```
            var folderName = "uploads";
            var PATH = HttpContext.Current.Server.MapPath("~/" + folderName);
            var rootUrl =
Request.RequestUri.AbsoluteUri.Replace(Request.RequestUri.AbsolutePath, String.Empty);
            if (Request.Content.IsMimeMultipartContent())
            {
                var streamProvider = new CustomMultipartFormDataStreamProvider(PATH);
                var task =
Request.Content.ReadAsMultipartAsync(streamProvider).ContinueWith<IEnumerable<FileDesc>>
(t =>
                {

                    if (t.IsFaulted || t.IsCanceled)
                    {
                        throw new
HttpResponseException(HttpStatusCode.InternalServerError);
                    }

                    var fileInfo = streamProvider.FileData.Select(i =>
                    {
                        var info = new FileInfo(i.LocalFileName);
                        return new FileDesc(info.Name, rootUrl + "/" + folderName + "/"
+ info.Name, info.Length / 1024);
                    });
                    return fileInfo;
                });

                return task;
            }
            else
            {
                throw new
HttpResponseException(Request.CreateResponse(HttpStatusCode.NotAcceptable, "This
request is not properly formatted"));
            }
        }
    }

    public class CustomMultipartFormDataStreamProvider :
MultipartFormDataStreamProvider
    {
        public CustomMultipartFormDataStreamProvider(string path)
            : base(path)
        { }
```

```
        public override string
GetLocalFileName(System.Net.Http.Headers.HttpContentHeaders headers)
        {
            var name = !string.IsNullOrWhiteSpace(headers.ContentDisposition.FileName)
? headers.ContentDisposition.FileName : "NoName";
            return name.Replace("\"", string.Empty);
        }
    }
```

MultipartFormDataStreamProvider

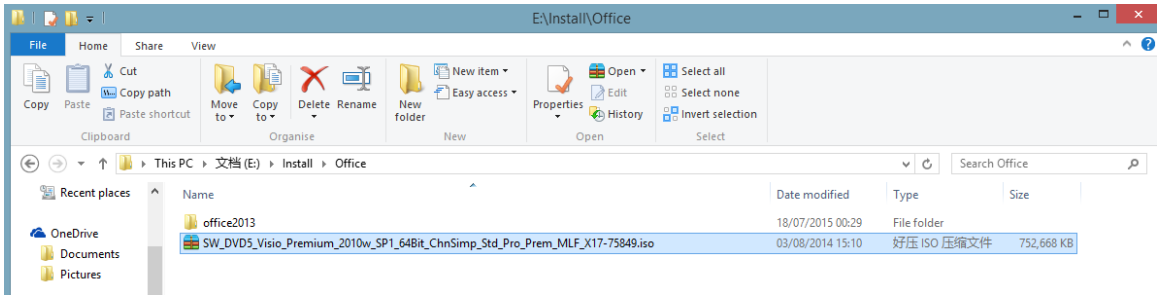
还有各种Provider，这样我们就可以有很多种处理方式，去掉WebAPI自带的Buffer，我们自己来处理。

MultipartFormDataRemoteStreamProvider 例如这个Provider，是抽象类，需要自己继承，设想如果我们要把这个上传的文件传到另外的地方去，例如Azure上，我们是不是可以直接架设管道，用流的方式。

下次介绍Provider。

最后还是上图吧，without picture I say ge hair?

准备工作，要上传的文件，才装了系统，就用这个吧



700多MB够了吧，还要怎样?

准备客户端，用网页比较方便，直接看进度

```
<h2>API Upload</h2>
<form name="apiForm" method="post" enctype="multipart/form-data"
action="http://localhost:15068/test/upload">
<div>
<label for="apifiles">Select a File</label>
<input name="apifiles" type="file" />
</div>
<div>
```

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

* *

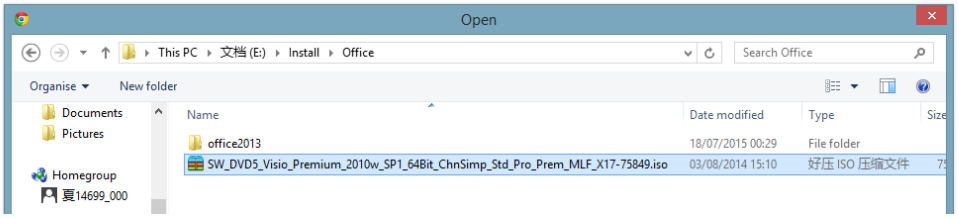
</form>

那就开始吧

API Upload

Select a File 选择文件 未选择任何文件

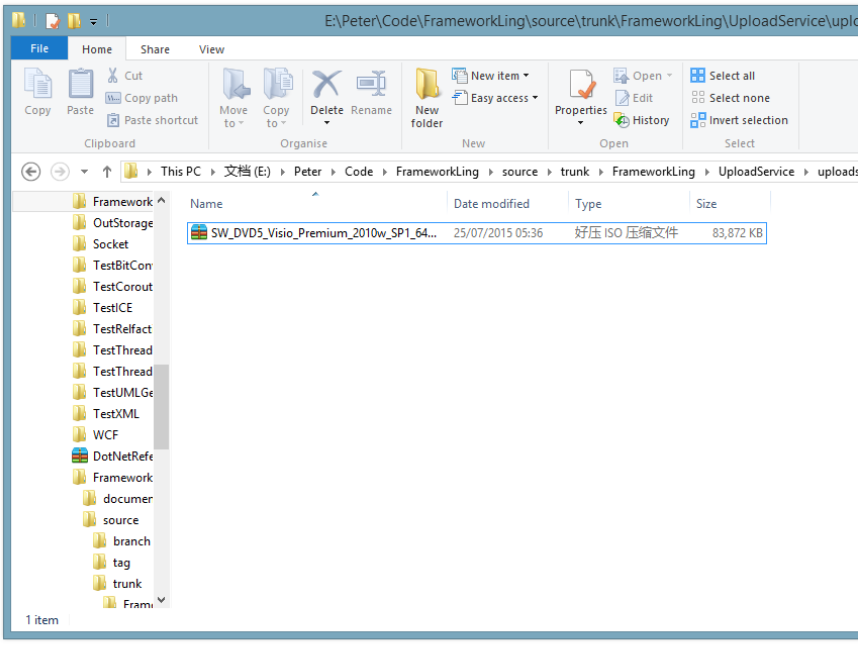
Upload



API Upload

Select a File 选择文件 SW_DVD5_Vis...-75849.iso

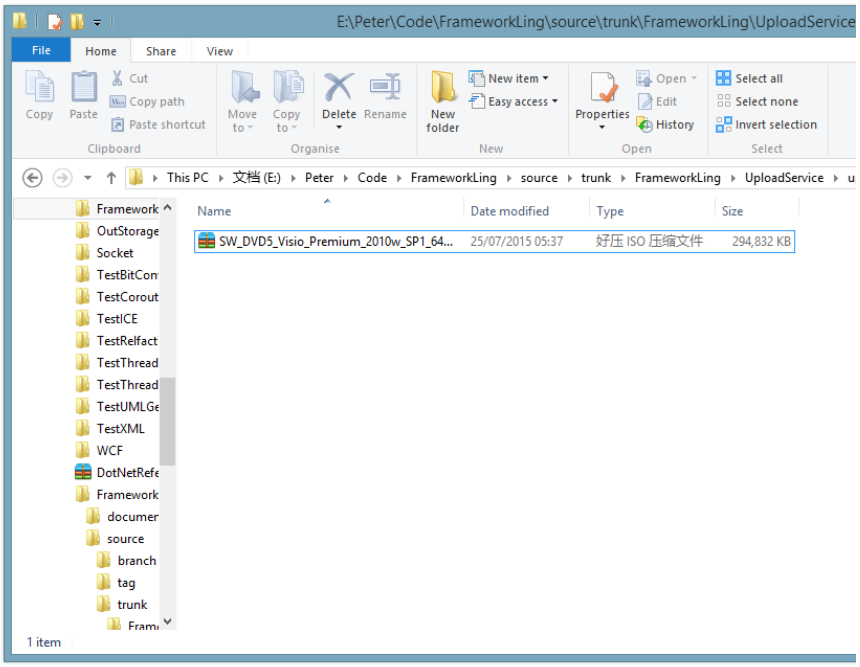
Upload



正在上传 (11%)...

API Upload

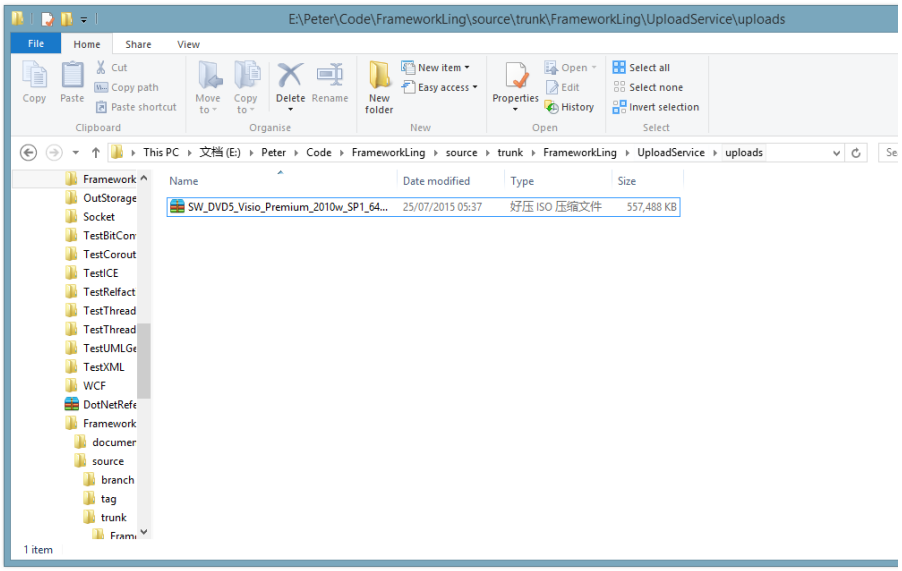
Select a File 选择文件 SW_DVD5_Vis...-75849.iso
Upload



E在上传 (39%)...

API Upload

Select a File 选择文件 SW_DVD5_Vis...-75849.iso
Upload



正在上传 (74%)...

至于内存的占用，既然是流，就不用多说了。不会有很大起伏的，就不贴图了，已经占了很大版面了。

PS：在学习的过程中，还遇到另外一种解决方案，是否还记得ASP.NET中接收上传文件的SaveAS方法，是的，用这个方法的主人也是可以接收文件的，它会干一个有趣的事情，不用内存作为Buffer，而是用硬盘，并且IIS在关闭或者隔一段时间后会干掉这个硬盘Buffer，后续的流程还是一样。对了，这是在IIS上测试的，本地宿主没试过。