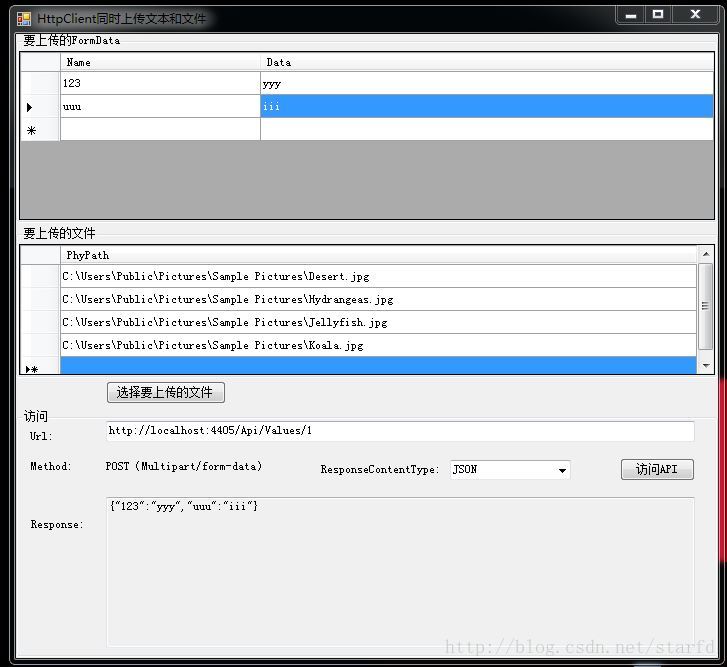
<http://blog.csdn.net/starfd/article/details/45393089>

**WebAPI通过multipart/form-data方式同时上传文件以及数据（含HttpClient上传Demo）**

简单的Demo，用于了解WebAPI如何同时接收文件及数据，同时提供HttpClient模拟如何同时上传文件和数据的Demo，下面是HttpClient上传的Demo界面



1、HttpClient部分：

HttpClient通过PostAsync提交数据时，第二个请求参数为抽象类HttpContent，当前我们需要通过multipart/form-data的方式模拟请求，multipart对应的请求HttpContent为MultipartContent及其子类MultipartFormDataContent，按名字明显可以看出MultipartFormDataContent对应multipart/form-data，MultipartFormDataContent可以通过Add方法添加具体的HttpContent，这里当然是添加ByteArrayContent了

下面是分别获取文件及键值对集合对应ByteArrayContent集合的代码

**[csharp]** [view plain](http://blog.csdn.net/starfd/article/details/45393089?readlog) [copy](http://blog.csdn.net/starfd/article/details/45393089?readlog)

1. /// <summary>
2. /// 获取文件集合对应的ByteArrayContent集合
3. /// </summary>
4. /// <param name="files"></param>
5. /// <returns></returns>
6. private List<ByteArrayContent> GetFileByteArrayContent(HashSet<string> files)
7. {
8. List<ByteArrayContent> list = new List<ByteArrayContent>();
9. foreach (var file in files)
10. {
11. var fileContent = new ByteArrayContent(File.ReadAllBytes(file));
12. fileContent.Headers.ContentDisposition = new ContentDispositionHeaderValue("attachment")
13. {
14. FileName = Path.GetFileName(file)
15. };
16. list.Add(fileContent);
17. }
18. return list;
19. }
20. /// <summary>
21. /// 获取键值集合对应的ByteArrayContent集合
22. /// </summary>
23. /// <param name="collection"></param>
24. /// <returns></returns>
25. private List<ByteArrayContent> GetFormDataByteArrayContent(NameValueCollection collection)
26. {
27. List<ByteArrayContent> list = new List<ByteArrayContent>();
28. foreach (var key in collection.AllKeys)
29. {
30. var dataContent = new ByteArrayContent(Encoding.UTF8.GetBytes(collection[key]));
31. dataContent.Headers.ContentDisposition = new ContentDispositionHeaderValue("attachment")
32. {
33. Name = key
34. };
35. list.Add(dataContent);
36. }
37. return list;
38. }

然后提交Api部分的代码如下（如需完整代码，请至底部点击源代码下载链接）

**[csharp]** [view plain](http://blog.csdn.net/starfd/article/details/45393089?readlog) [copy](http://blog.csdn.net/starfd/article/details/45393089?readlog)

1. using (HttpClient client = new HttpClient())
2. {
3. client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("text/" + this.cmbResponseContentType.Text.ToLower()));//设定要响应的数据格式
4. using (var content = new MultipartFormDataContent())//表明是通过multipart/form-data的方式上传数据
5. {
6. var formDatas = this.GetFormDataByteArrayContent(this.GetNameValueCollection(this.gv\_FormData));//获取键值集合对应的ByteArrayContent集合
7. var files = this.GetFileByteArrayContent(this.GetHashSet(this.gv\_File));//获取文件集合对应的ByteArrayContent集合
8. Action<List<ByteArrayContent>> act = (dataContents) =>
9. {//声明一个委托，该委托的作用就是将ByteArrayContent集合加入到MultipartFormDataContent中
10. foreach (var byteArrayContent in dataContents)
11. {
12. content.Add(byteArrayContent);
13. }
14. };
15. act(formDatas);//执行act
16. act(files);//执行act
17. try
18. {
19. var result = client.PostAsync(this.txtUrl.Text, content).Result;//post请求
20. this.txtResponse.Text = result.Content.ReadAsStringAsync().Result;//将响应结果显示在文本框内
21. }
22. catch (Exception ex)
23. {
24. this.txtResponse.Text = ex.ToString();//将异常信息显示在文本框内
25. }
26. }
27. }

2、WebAPI部分

其实WebAPI这部分真的没什么，完全是参考了国外大牛的代码，不过某些不明了的地方在方法内有备注，有时间会去研究下如何才能实现无需保存文件至硬盘，即可获取相应的数据流

**[csharp]** [view plain](http://blog.csdn.net/starfd/article/details/45393089?readlog) [copy](http://blog.csdn.net/starfd/article/details/45393089?readlog)

1. [HttpPost]
2. public async Task<Dictionary<string, string>> Post(int id = 0)
3. {
5. if (!Request.Content.IsMimeMultipartContent())
6. {
7. throw new HttpResponseException(HttpStatusCode.UnsupportedMediaType);
8. }
9. Dictionary<string, string> dic = new Dictionary<string, string>();
10. string root = HttpContext.Current.Server.MapPath("~/App\_Data");//指定要将文件存入的服务器物理位置
11. var provider = new MultipartFormDataStreamProvider(root);
12. try
13. {
14. // Read the form data.
15. await Request.Content.ReadAsMultipartAsync(provider);
17. // This illustrates how to get the file names.
18. foreach (MultipartFileData file in provider.FileData)
19. {//接收文件
20. Trace.WriteLine(file.Headers.ContentDisposition.FileName);//获取上传文件实际的文件名
21. Trace.WriteLine("Server file path: " + file.LocalFileName);//获取上传文件在服务上默认的文件名
22. }//TODO:这样做直接就将文件存到了指定目录下，暂时不知道如何实现只接收文件数据流但并不保存至服务器的目录下，由开发自行指定如何存储，比如通过服务存到图片服务器
23. foreach (var key in provider.FormData.AllKeys)
24. {//接收FormData
25. dic.Add(key, provider.FormData[key]);
26. }
27. }
28. catch
29. {
30. throw;
31. }
32. return dic;
33. }

[源代码下载](http://download.csdn.net/detail/starfd/8648853)，运行Demo时请先调试服务端，然后开启客户端，如果缺少HttpClient对应的dll，请通过NuGet下载

<http://blog.csdn.net/u014742815/article/details/52537093>

<div class="jumbotron">

<h1>Web Api下载文件示例</h1>

<p><a href="http://localhost:60560/api/download/get\_demo\_file" class="btn btn-primary btn-lg">下载示例文件 &raquo;</a></p>

</div>

* 1
* 2
* 3
* 4
* 5



using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Net.Http.Headers;

using System.Web.Http;

namespace DownloadFileFromWebApi.Controllers

{

/// <summary>

/// 实现的方法很简单，其中就是读取服务器的指定路径文件流，将其做为返回的HttpResponseMessage的Content

/// </summary>

[RoutePrefix("api/download")]

public class DownloadController : ApiController

{

[HttpGet,Route("get\_demo\_file")]

public HttpResponseMessage GetFileFromWebApi()

{

try

{

var FilePath = System.Web.Hosting.HostingEnvironment.MapPath(@"~/download/EditPlus64\_xp85.com.zip");

var stream = new FileStream(FilePath, FileMode.Open);

HttpResponseMessage response = new HttpResponseMessage(HttpStatusCode.OK);

response.Content = new StreamContent(stream);

response.Content.Headers.ContentType = new MediaTypeHeaderValue("application/octet-stream");

response.Content.Headers.ContentDisposition = new ContentDispositionHeaderValue("attachment")

{

FileName = "Wep Api Demo File.zip"

};

return response;

}

catch

{

return new HttpResponseMessage(HttpStatusCode.NoContent);

}

}

}

}