Download file with WebClient or HttpClient?

Asked 3 years, 6 months ago Active 11 days ago Viewed 78k times

44

13

I am trying to download file from a URL and I have to choose between WebClient and HttpClient. I have referenced this article and several other articles on the internet. Everywhere, it is suggested to go for HttpClient due to its great async support and other .Net 4.5 privileges. But I am still not totally convinced and need more inputs.

I am using below code to download file from internet:

WebClient:

```
WebClient client = new WebClient();
client.DownloadFile(downloadUrl, filePath);
```

HttpClient:

```
using (HttpClient client = new HttpClient())
{
    using (HttpResponseMessage response = await client.GetAsync(url))
    using (Stream streamToReadFrom = await response.Content.ReadAsStreamAsync())
    {
    }
}
```

From my perspective, I can see only one disadvantage in using WebClient, that would be the non async call, blocking the calling thread. But what if I am not worried about the blocking of thread or use `client.DownloadFileAsync()` to leverage the async support?

On the other hand, if I use HttpClient, ain't I loading every single byte of a file into memory and then writing it to a local file? If the file size is too large, won't memory overhead be expensive? Which could be avoided if we use WebClient, since it will directly write to local file and not consume system memory.

So, if performance is my utter priority, which approach should I use for download? I would like to be clarified if my above assumption is wrong, and I am open to alternate approach as well.

c# .net rest httpclient webclient

Share Edit Follow

asked Aug 16 '17 at 10:44
Saket
2,868 3 22 46

Does stackoverflow.com/questions/37799419/… help? – mjwills Aug 16 '17 at 10:47

Also see codereview.stackexchange.com/questions/69950/… . – mjwills Aug 16 '17 at 10:48

3 "disadvantage in using WebClient...the non async call, blocking the calling thread" So use

`DownloadFileAsync` . – Kenneth K. Aug 7 '18 at 19:34

2 WebClient is obsolete sincd 2012 and the two snippets are doing different things. You can use HttpClient.GetStreamAsync to get a stream to the file in one line and then use `.CopyToAsync()` to copy the stream's contents to a file stream – Panagiotis Kanavos Feb 1 '19 at 7:50

1 @KennethK. you probably mean DownloadFileTaskAsync. The older DownloadFileAsync uses events to notify that a download completed, it's not asynchronous in the sense used nowadays – Panagiotis Kanavos Feb 1 '19 at 7:54

7 Answers

Active Oldest Votes

22

You can do it natively with .Net 4.5+. I tried doing it your way and then I just found a method in Intellisense that seemed to make sense.

https://docs.microsoft.com/en-us/dotnet/api/system.io.stream.copytoasync?view=netframework-4.7.2

```
uri = new Uri(generatePdfsRetrieveUrl + pdfGuid + ".pdf");
HttpClient client = new HttpClient();
var response = await client.GetAsync(uri);
using (var fs = new FileStream(
    HostingEnvironment.MapPath(string.Format("~/Downloads/{0}.pdf", pdfGuid)),
    FileMode.CreateNew))
{
    await response.Content.CopyToAsync(fs);
}
```

Share Edit Follow edited Jan 6 at 15:14 answered Feb 1 '19 at 7:43

Bluebaron
1,593 1 18 29

I ended up using this, note `HostingEnvironment.MapPath(string.Format("~/Downloads/{0}.pdf", pdfGuid)),` can be replaced with an arbitrary path. – Felipe Gutierrez Mar 18 '20 at 22:09

Shouldn't `HttpClient client = new HttpClient();` and `var response = await client.GetAsync(uri);` be in using statements? It says they inherit IDisposable. – Trisped Dec 29 '20 at 0:42

@Trisped You can, but depending on your usage you should consider that when a client is disposed the port will be left in a `TIME_WAIT` state. See here – Mitchell Wright Feb 19 at 14:22

Doesn't the C# 8.0 automatically dispose of disposable objects when they go out of scope? – Bluebaron Feb 19 at 16:33

12

Here is my approach.

If you are calling a WebApi to get a file, then from a controller method you can use HttpClient GET request and return file stream using FileStreamResult return type.

```
public async Task<ActionResult> GetAttachment(int FileID)
{
    UriBuilder uriBuilder = new UriBuilder();
    uriBuilder.Scheme = "https";
    uriBuilder.Host = "api.example.com";
```

```
        var Path = "/files/download";
        uriBuilder.Path = Path;
        using (HttpClient client = new HttpClient())
        {
            client.BaseAddress = new Uri(uriBuilder.ToString());
            client.DefaultRequestHeaders.Accept.Clear();
            client.DefaultRequestHeaders.Add("authorization", access_token); //if any
            client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
            HttpResponseMessage response = await client.GetAsync(uriBuilder.ToString());

                if (response.IsSuccessStatusCode)
                {
                    System.Net.Http.HttpContent content = response.Content;
                    var contentStream = await content.ReadAsStreamAsync(); // get the
actual content stream
                    return File(contentStream, content_type, filename);
                }
                else
                {
                    throw new FileNotFoundException();
                }
        }
    }
```

Share Edit Follow

answered Aug 7 '18 at 19:30
Ingale88s
228 2 7

7 you should not use the HttpClient as a Disposable object although it is, you will get socket exhaustion if you have many requests. Use the httpClient has a static instance instead, lots of articles online covering this problem. – Rui Lima Dec 21 '19 at 10:47

It should also not be used as a `static` instance @RuiLima, but managed through the `HttpClientFactory` . The static instance can also cause problems as it is never refreshed. – julealgon Jan 28 at 18:20

3

Here’s one way to use it to download a URL and save it to a file: (I am using windows 7, therefore no WindowsRT available to me, so I’m also using System.IO.)

```
public static class WebUtils
{
    private static Lazy<IWebProxy> proxy = new Lazy<IWebProxy>(() =>
string.IsNullOrEmpty(Settings.Default.WebProxyAddress) ? null : new WebProxy { Address
= new Uri(Settings.Default.WebProxyAddress), UseDefaultCredentials = true });

    public static IWebProxy Proxy
    {
        get { return WebUtils.proxy.Value; }
    }

    public static Task DownloadAsync(string requestUri, string filename)
    {
        if (requestUri == null)
            throw new ArgumentNullException(“requestUri”);

        return DownloadAsync(new Uri(requestUri), filename);
    }

    public static async Task DownloadAsync(Uri requestUri, string filename)
    {
        if (filename == null)
            throw new ArgumentNullException("filename");

        if (Proxy != null)
            WebRequest.DefaultWebProxy = Proxy;

        using (var httpClient = new HttpClient())
        {
            using (var request = new HttpRequestMessage(HttpMethod.Get, requestUri))
            {
                using (Stream contentStream = await (await
httpClient.SendAsync(request)).Content.ReadAsStreamAsync(), stream = new
FileStream(filename, FileMode.Create, FileAccess.Write, FileShare.None,
Constants.LargeBufferSize, true))
                {
                    await contentStream.CopyToAsync(stream);
                }
            }
        }
    }
}
```

Note that code is saving the address of the proxy server I use (at work) in a setting, and using that if such setting is specified. Otherwise, it should tell you all you need to know regarding using the HttpClient beta to download and save a file.

Share Edit Follow

edited Apr 4 '19 at 13:35
Ian Kemp
23.4k 14 97 119

answered Aug 16 '17 at 10:51
Nirzar
131 1 10

2 codereview.stackexchange.com/questions/69950/… may be worth a read. – mjwills Aug 16 '17 at 11:05

3 Most of this code isn't related to the question. HttpClient instances *shouldn't* be disposed immediately either. – Panagiotis Kanavos Feb 1 '19 at 7:56

2

If you want (or have) to do this synchronously, but using the nice HttpClient class, then there's this simple approach:

```
string requestString = @"https://example.com/path/file.pdf";

var GetTask = httpClient.GetAsync(requestString);
GetTask.Wait(WebCommsTimeout); // WebCommsTimeout is in milliseconds

if (!GetTask.Result.IsSuccessStatusCode)
{
    // write an error
    return;
}

using (var fs = new FileStream(@"c:\path\file.pdf", FileMode.CreateNew))
{
    var ResponseTask = GetTask.Result.Content.CopyToAsync(fs);
    ResponseTask.Wait(WebCommsTimeout);
}
```

Share Edit Follow

edited Feb 15 at 6:13
Muflix
4,136 8 51 109

answered Sep 23 '19 at 20:52
user11913024

2

For code being called repeatedly, you **do not** want to put `HttpClient` in a `using` block (it will leave hanging ports open)

For downloading a file with HttpClient, I found this extension method which seemed like a good and reliable solution to me:

```
public static class HttpContentExtensions
{
    public static Task ReadAsFileAsync(this HttpContent content, string filename, bool
overwrite)
    {
        string pathname = Path.GetFullPath(filename);
        if (!overwrite && File.Exists(filename))
        {
            throw new InvalidOperationException(string.Format("File {0} already
exists.", pathname));
        }

        FileStream fileStream = null;
        try
        {
            fileStream = new FileStream(pathname, FileMode.Create, FileAccess.Write,
FileShare.None);
            return content.CopyToAsync(fileStream).ContinueWith(
                (copyTask) =>
                {
                    fileStream.Close();
                });
        }
        catch
        {
            if (fileStream != null)
            {
                fileStream.Close();
            }

            throw;
        }
    }
}
```

Share Edit Follow

answered Sep 18 '18 at 22:08

Thymine

7,619 1 28 42

0

To use HttpClient on my existing code that used WebClient, I wrote a small extension method to use it on the same way I used `DownloadFileTaskAsync` on my code.

```
using (var client = new System.Net.Http.HttpClient()) // WebClient
{
    var fileName = @"C:\temp\imgd.jpg";
    var uri = new Uri("https://yourwebsite.com/assets/banners/Default.jpg");

    await client.DownloadFileTaskAsync(uri, fileName);
}
```

To use it we can have this extension method:

```
public static class HttpClientUtils
{
    public static async Task DownloadFileTaskAsync(this HttpClient client, Uri uri,
string FileName)
    {
        using (var s = await client.GetStreamAsync(uri))
        {
            using (var fs = new FileStream(FileName, FileMode.CreateNew))
            {
                await s.CopyToAsync(fs);
            }
        }
    }
}
```

Share Edit Follow

answered Feb 19 at 0:59

Tony
12.9k 11 64 112

-1

```
HttpClient _client=new HttpClient();
byte[] buffer = null;
try
{
   HttpResponseMessage task = await _client.GetAsync("https://**FILE_URL**");
   Stream task2 = await task.Content.ReadAsStreamAsync();
   using (MemoryStream ms = new MemoryStream())
   {
     await task2.CopyToAsync(ms);
     buffer = ms.ToArray();
   }
   File.WriteAllBytes("C:/**PATH_TO_SAVE**", buffer);
}
catch
{

}
```

Share Edit Follow

answered Jul 14 '20 at 11:57

Ernest Rutherford
99 3