Getting an UTF-8 response with httpclient in Windows Store apps

Asked 6 years, 11 months ago Active 2 years, 2 months ago Viewed 27k times

14

2

I'm building a Windows Store app, but I'm stuck at getting a UTF-8 response from an API.

This is the code:

```
using (HttpClient client = new HttpClient())
{
    Uri url = new Uri(BaseUrl + "/me/lists");

    HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, url);
    request.Headers.Add("Accept", "application/json");
    HttpResponseMessage response = await client.SendRequestAsync(request);
    response.EnsureSuccessStatusCode();

    string responseString = await response.Content.ReadAsStringAsync();

    response.Dispose();
}
```

The `reponseString` always contains strange characters which should be accents like *é*, and I tried using a stream, but the API I found in some examples don't exist in Windows RT.

Edit: improved code, still same problem.

c# .net utf-8 character-encoding windows-runtime

Share Edit Follow

edited Dec 17 '18 at 0:09
PoLáKoSz
355 1 6 7

asked Mar 26 '14 at 1:00
Sam Debruyn
788 1 7 18

Are you completely sure API you are calling handles non-ASCII text correctly? (hard to guess from your code what you are calling) – Alexei Levenkov Mar 26 '14 at 1:05

It's not clear at all what the problem is you're describing. Are you expecting a UTF-8 response? Is the server sending one? Can you hit the web service directly using Fiddler to see what it's returning? – WiredPrairie Mar 26 '14 at 1:38

Is the Content-Type header of the response set to the correct value? Have you tried the method `_client.GetStringAsync(url)` ? – cremor Mar 26 '14 at 8:28

@AlexeiLevenkov Yes, in the Chrome REST Console I get a good UTF-8 response, accents are shown where they should be. – Sam Debruyn Mar 26 '14 at 8:58

@WiredPrairie I don't know how to use Fiddler for HTTPS requests. But in the Chrome REST Console I see an UTF-8 response. – Sam Debruyn Mar 26 '14 at 9:00

7 Answers

Active | Oldest | Votes

Instead of using `response.Content.ReadAsStringAsync()` directly you could use

27

`response.Content.ReadAsBufferAsync()` pointed by @Kiewic as follows:

```
var buffer = await response.Content.ReadAsBufferAsync();
var byteArray = buffer.ToArray();
var responseString = Encoding.UTF8.GetString(byteArray, 0, byteArray.Length);
```

This is working in my case and I guess that using UTF8 should solve most of the issues. Now go figure why there is no way to do this using `ReadAsStringAsync` :)

Share Edit Follow

edited Aug 10 '15 at 15:32

answered Jan 25 '15 at 0:21

Camilo Martinez
1,433 1 20 26

is that a typo? It seems like the method name is just repeated: "Instead of using response.Content.ReadAsStringAsync() directly you could use response.Content.ReadAsStringAsync()" – superjos Aug 10 '15 at 13:10

@superjos You're right, there was a typo, the method is ReadAsBufferAsync. I just fixed it. – Camilo Martinez Aug 10 '15 at 15:33

I'm having this problem, but this doesn't fix my problem. My code works fine in .NET Core, but I am getting garbled characters in UWP. – Christian Findlay Jan 15 '18 at 5:53

4 ReadAsBufferAsync doesn't exist. maybe ReadAsByteArrayAsync? – syonip Apr 15 '19 at 7:45

4

Solved it like this:

```
using (HttpClient client = new HttpClient())
    {
        using (HttpResponseMessage response = client.GetAsync(url).Result)
            {
                var byteArray = response.Content.ReadAsByteArrayAsync().Result;
                var result = Encoding.UTF8.GetString(byteArray, 0, byteArray.Length);
                return result;
            }
    }
```

Share Edit Follow

answered May 15 '17 at 13:40

Ogglas
35.3k 19 185 249

3

I like El Marchewko's approach of using an extension, but the code did not work for me. This did:

```
using System.IO;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;

namespace WannaSport.Data.Integration
{
    public static class HttpContentExtension
    {
        public static async Task<string> ReadAsStringUTF8Async(this HttpContent
```

```
content)
        {
            return await content.ReadAsStringAsync(Encoding.UTF8);
        }

        public static async Task<string> ReadAsStringAsync(this HttpContent content,
Encoding encoding)
        {
            using (var reader = new StreamReader((await content.ReadAsStreamAsync()),
encoding))
            {
                return reader.ReadToEnd();
            }
        }
    }
}
```

Share Edit Follow

answered Mar 31 '16 at 11:38

pius
1,796 19 22

I'm getting the same issue in UWP. My code works fine in .NET Core, but I get totally garbled characters in UWP. I tried this exact code and it didn't work. – Christian Findlay Jan 15 '18 at 5:56

1

The `HttpClient` doesn't give you a lot of flexibility.

You can use a `HttpWebRequest` instead and get the raw bytes from the response using `HttpWebResponse.GetResponseStream()` .

Share Edit Follow

answered Mar 26 '14 at 9:59

BW

BartW
11 2

1 What about `Stream stream = await response.Content.ReadAsStreamAsync()` – kiewic Mar 26 '14 at 17:51

1

Perhaps the problem is that the response is zipped. If the content type is gzip, you will need decompress the response in to a string. Some servers do this to save bandwidth which is normally fine. In .NET Core and probably .NET Framework, this will automatically unzip the response. But this does not work in UWP. This seems like a glaring bug in UWP to me.

```
string responseString = await response.Content.ReadAsStringAsync();
```

This thread gives a clear example of how to decompress the response:

Compression/Decompression string with C#

Share Edit Follow

answered Jan 15 '18 at 6:35

Christian Findlay
4,496 1 30 71

This solved it for me. Just to be sure before trying to decompress the response I disabled the "Accept-Encoding" header by commenting-out `HttpClient.DefaultRequestHeaders.Add("*Accept-Encoding*", "gzip, deflate, br");` and it worked fine after that. – Anthony Walsh Mar 17 '19 at 5:39

0

Can't comment yet, so I'll have to add my thoughts here.

You could try to use `_client.GetStringAsync(url)` as @cremor suggested, and set your authentication headers using the `_client.DefaultRequestHeaders` property. Alternatively, you could also try to use the `ReadAsByteArrayAsync` method on the `response.Content` object and use `System.Text.Encoding` to decode that byte array to a UTF-8 string.

Share Edit Follow

answered Mar 26 '14 at 9:19

Wesley Cabus
111 1 1 7

ReadAsByteArrayAsync is not available in WinRT – Sam Debruyn Mar 26 '14 at 9:56

@SamuelDebruyn `response.Content.ReadAsByteArrayAsync()` is supported in WinRT. – kiewic Mar 26 '14 at 17:58

@Kiewic The only 3 options I get when I start typing response.Content.Read... are `ReadAsBufferAsync`, `ReadAsInputStreamAsync` and `ReadAsStringAsync` – Sam Debruyn Mar 27 '14 at 14:08

2 Oh you are using Wondows.Web.Http.HttpClient, then ReadAsBuffer should work. you can convert a butter to a byte array. – kiewic Mar 30 '14 at 19:31

0

My approach using an Extension:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Windows.Web.Http;

namespace yourfancyNamespace
{
    public static class IHttpContentExtension
    {
        public static async Task<string> ReadAsStringUTF8Async(this IHttpContent
content)
        {
            return await content.ReadAsStringAsync(Encoding.UTF8);
        }
        public static async Task<string> ReadAsStringAsync(this IHttpContent
content, Encoding encoding)
        {
            using (TextReader reader = new StreamReader((await
content.ReadAsInputStreamAsync()).AsStreamForRead(), encoding))
            {
                return reader.ReadToEnd();
            }
        }
    }
}
```

Share Edit Follow

answered Mar 25 '15 at 14:00

Jens Marchewka

1,342 1 11 22