Is there a way to force ASP.NET Web API to return plain text?

Ask Question

I need to get a response back in plain text from a ASP.NET Web API controller.

I have tried do a request with `Accept: text/plain` but it doesn't seem to do the trick. Besides, the request is external and out of my control. What I would accomplish is to mimic the old ASP.NET way:

```
context.Response.ContentType = "text/plain";
context.Response.Write("some text);
```

Any ideas?

EDIT, solution: Based on Aliostad's answer, I added the WebAPIContrib text formatter, initialized it in the Application_Start:

```
  config.Formatters.Add(new PlainTextFormatter());
```

and my controller ended up something like:

```
[HttpGet, HttpPost]
public HttpResponseMessage GetPlainText()
{
  return ControllerContext.Request.CreateResponse(HttpStatusCode.OK, "Test data",
"text/plain");
}
```

asp.net asp.net-web-api

edited Jul 20 '12 at 15:39

asked Jul 20 '12 at 14:49

Magnus Johansson
20.6k 15 80 139

6 Answers

Hmmm... I don't think you need to create a custom formatter to make this work. Instead return the content like this:

```
    [HttpGet]
    public HttpResponseMessage HelloWorld()
    {
        string result = "Hello world! Time is: " + DateTime.Now;
        var resp = new HttpResponseMessage(HttpStatusCode.OK);
        resp.Content = new StringContent(result, System.Text.Encoding.UTF8,
"text/plain");
        return resp;
    }
```

This works for me without using a custom formatter.

If you explicitly want to create output and override the default content negotiation based on Accept headers you won't want to use `Request.CreateResponse()` because it forces the mime type.

Instead explicitly create a new `HttpResponseMessage` and assign the content manually. The example above uses `StringContent` but there are quite a few other content classes available to return data from various .NET data types/structures.

edited Dec 11 '15 at 23:47

answered Oct 23 '12 at 10:04

Rick Strahl
9,684 7 55 85

1 This is in fact the solution I went for because my API would be returning JSON objects to 99% of all methods, only a few (very few) methods would need plain string responses (and for many of those I use a MemoryStream to return data directly in the response so it was a non-issue.) Only in 2 or 3 methods did I return a .NET string, and it was being returned as a JSON string. Your answer, IMHO, is the KISS response for this problem (although it is not

Nachiket Mehta May 12 '15 at 22:05

@JavascriptEnthusiast - HttpContext.Current is null most likely because you are self-hosting or running through the OWin stack without the System.Web pipeline. Unrelated to this solution though. – Rick Strahl Jul 5 '15 at 23:46

- **Please be careful** not to use context in ASP.NET Web API or you will sooner or later be sorry. Asynchronous nature of ASP.NET Web API makes using `HttpContext.Current` a liability.
- Use a plain text formatter and add to your formatters. There are dozens of them around. You could even write yours easily. WebApiContrib has one.
- You can force it by setting the content type header on `httpResponseMessage.Headers` to `text/plain` in your controller provided you have registered plain text formatter.

answered Jul 20 '12 at 15:19

Aliostad
66.8k 13 127 183

Don't worry, I neither implied nor intended to use the HttpContext object, I just added it to illustrate how one would do it in classic ASP.NET – Magnus Johansson Jul 20 '12 at 15:29

Well, waddayknow, I already had WebAPIContrib referenced, sometimes it's simple. – Magnus Johansson Jul 20 '12 at 15:40

@Magnus Sure. I in fact changed the wording after I read what I had written. But reading another answer made me stress that first point. – Aliostad Jul 20 '12 at 15:44

You are saying not to use HttpContext.Current, what are the alternatives? – surya Aug 9 '13 at 13:35

@spiderdevil yes, it is absolutely what I am saying. You should not need it, pass request/response/configuration directly. – Aliostad Aug 9 '13 at 16:29

If you are just looking for a simple plain/text formatter without adding additional dependencies, this should do the trick.

```
public class TextPlainFormatter : MediaTypeFormatter
{
    public TextPlainFormatter()
    {
        this.SupportedMediaTypes.Add(new MediaTypeHeaderValue("text/plain"));
    }

    public override bool CanWriteType(Type type)
    {
        return type == typeof(string);
    }

    public override bool CanReadType(Type type)
    {
        return type == typeof(string);
    }

    public override Task WriteToStreamAsync(Type type, object value, Stream stream,
HttpContentHeaders contentHeaders, TransportContext transportContext)
    {
        return Task.Factory.StartNew(() => {
            StreamWriter writer = new StreamWriter(stream);
            writer.Write(value);
            writer.Flush();
        });
    }

    public override Task<object> ReadFromStreamAsync(Type type, Stream stream,
HttpContentHeaders contentHeaders, IFormatterLogger formatterLogger)
    {
        return Task.Factory.StartNew(() => {
            StreamReader reader = new StreamReader(stream);
            return (object)reader.ReadToEnd();
        });
    }
}
```

Now you can pass string objects to

```
this.Request.CreateResponse(HttpStatusCode.OK, "some text", "text/plain");
```

edited Feb 13 '14 at 0:05

answered Oct 13 '12 at 7:35

Despertar

14.8k 5 58 71

When Accept: text/plain doesnt work, then there is no registered formatter for text mime types.

You can ensure that there is no formatters for specified mime type by getting list of all supported formatters from service configuration.

Create a very straightforward media type formatter that support text mime types.

http://www.asp.net/web-api/overview/formats-and-model-binding/media-formatters

answered Jul 20 '12 at 14:53

Regfor

6,550 1 25 44

Wish I could accepted your answer as well, the accepted answer saved me the trouble of writing my own formatter. +1 at least. – Magnus Johansson Jul 20 '12 at 15:49

For .net core:

```
[HttpGet("About")]
public ContentResult About()
{
    return Content("About text");
}
```

https://docs.microsoft.com/en-us/aspnet/core/mvc/models/formatting

answered Apr 4 '17 at 21:30

rook

1,123 2 9 28

An extension like the following one can reduce the number of lines and beautify your code:

```
public static class CommonExtensions
{
    public static HttpResponseMessage ToHttpResponseMessage(this string str)
    {
        var resp = new HttpResponseMessage(HttpStatusCode.OK)
        {
            Content = new StringContent(str, System.Text.Encoding.UTF8, "text/plain")
        };

        return resp;
    }
}
```

Now you can consume the defined extension in your `Web API` :

```
public class HomeController : ApiController
{
    [System.Web.Http.HttpGet]
    public HttpResponseMessage Index()
    {
        return "Salam".ToHttpResponseMessage();
    }
}
```

By routing `{DOMAIN}/api/Home/Index` you can see the following plain text:

> MyPlainTextResponse

answered Jan 4 at 22:11