How to enable CORS in ASP.net Core WebAPI

113

50

What I am trying to do

I have a backend ASP.Net Core Web API hosted on an Azure Free Plan (Source Code: https://github.com/killerrin/Portfolio-Backend).

I also have a Client Website which I want to make consume that API. The Client Application will not be hosted on Azure, but rather will be hosted on Github Pages or on another Web Hosting Service that I have access to. Because of this the domain names won't line up.

Looking into this, I need to enable CORS on the Web API side, however I have tried just about everything for several hours now and it is refusing to work.

How I have the Client Setup Its just a simple client written in React.js. I'm calling the APIs through AJAX in Jquery. The React site works so I know its not that. The Jquery API call works as I confirmed in Attempt 1. Here is how I make the calls

```
    var apiUrl =
"http://andrewgodfroyportfolioapi.azurewebsites.net/api/Authentication";
    //alert(username + "|" + password + "|" + apiUrl);
    $.ajax({
        url: apiUrl,
        type: "POST",
        data: {
            username: username,
            password: password
        },
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: function (response) {
            var authenticatedUser = JSON.parse(response);
            //alert("Data Loaded: " + authenticatedUser);
            if (onComplete != null) {
                onComplete(authenticatedUser);
            }
        },
        error: function (xhr, status, error) {
            //alert(xhr.responseText);
```

What I have tried

Attempt 1 - The 'proper' way

https://docs.microsoft.com/en-us/aspnet/core/security/cors

I have followed this tutorial on the Microsoft Website to a T, trying all 3 options of enabling it Globally in the Startup.cs, Setting it up on every controller and Trying it on every Action.

Following this method, the Cross Domain works, but only on a single Action on a single controller (POST to the AccountController). For everything else, the `Microsoft.AspNetCore.Cors` middleware refuses to set the headers.

I installed `Microsoft.AspNetCore.Cors` through NUGET and the version is `1.1.2`

Here is how I have it setup in Startup.cs

```
    // This method gets called by the runtime. Use this method to add services to the
container.
    public void ConfigureServices(IServiceCollection services)
    {
        // Add Cors
        services.AddCors(o => o.AddPolicy("MyPolicy", builder =>
        {
            builder.AllowAnyOrigin()
                   .AllowAnyMethod()
                   .AllowAnyHeader();
        }));

        // Add framework services.
        services.AddMvc();
        services.Configure<MvcOptions>(options =>
        {
            options.Filters.Add(new CorsAuthorizationFilterFactory("MyPolicy"));
        });

        ...
```

```
        ILoggerFactory loggerFactory)
        {
            loggerFactory.AddConsole(Configuration.GetSection("Logging"));
            loggerFactory.AddDebug();

            // Enable Cors
            app.UseCors("MyPolicy");

            //app.UseMvcWithDefaultRoute();
            app.UseMvc();

            ...
            ...
            ...
        }
```

As you can see, I am doing everything as told. I add Cors before MVC both times, and when that didn't work I attempted putting `[EnableCors("MyPolicy")]` on every controller as so

```
[Route("api/[controller]")]
[EnableCors("MyPolicy")]
public class AdminController : Controller
```

Attempt 2 - Brute Forcing it

https://andrewlock.net/adding-default-security-headers-in-asp-net-core/

After several hours of trying on the previous attempt, I figured I would try to bruteforce it by trying to set the headers manually, forcing them to run on every response. I did this following this tutorial on how to manually add headers to every response.

These are the headers I added

```
.AddCustomHeader("Access-Control-Allow-Origin", "*")
.AddCustomHeader("Access-Control-Allow-Methods", "*")
.AddCustomHeader("Access-Control-Allow-Headers", "*")
.AddCustomHeader("Access-Control-Max-Age", "86400")
```

```
.AddCustomHeader("Access-Control-Allow-Headers", "X-PINGOTHER, Host, User-Agent, Accept,
Accept: application/json, application/json, Accept-Language, Accept-Encoding, Access-
Control-Request-Method, Access-Control-Request-Headers, Origin, Connection, Content-
Type, Content-Type: application/json, Authorization, Connection, Origin, Referer")
```

With this method, the Cross Site headers are being properly applied and they show up in my developer console and in Postman. The problem however is that while it passes the `Access-Control-Allow-Origin` check, the webbrowser throws a hissy fit on (I believe) `Access-Control-Allow-Headers` stating `415 (Unsupported Media Type)`

So the brute force method doesn't work either

Finally

Has anyone gotten this to work and could lend a hand, or just be able to point me in the right direction?

EDIT

So to get the API calls to go through, I had to stop using JQuery and switch to a Pure Javascript `XMLHttpRequest` format.

Attempt 1

I managed to get the `Microsoft.AspNetCore.Cors` to work by following MindingData's answer, except within the `Configure` Method putting the `app.UseCors` before `app.UseMvc` .

In addition, when mixed with the Javascript API Solution `options.AllowAnyOrigin()` for wildcard support began to work as well.

Attempt 2

So I have managed to get Attempt 2 (brute forcing it) to work... with the only exception that the Wildcard for `Access-Control-Allow-Origin` doesn't work and as such I have to manually set the domains that have access to it.

Its obviously not ideal since I just want this WebAPI to be wide opened to everyone, but it atleast works for me on a separate site, which means it's a start

```
    .AddCustomHeader("Access-Control-Allow-Headers", "X-PINGOTHER, Content-Type,
Authorization"));
```

c# rest asp.net-core cors cross-domain

edited Jan 19 at 16:28

asked Jun 6 '17 at 0:14

killerrin

692 2 6 7

2 For your `415 (Unsupported Media Type)` issue, set a `Content-Type` request header to `application/json` . – Technetium Jun 6 '17 at 0:54

2 Thanks for spending the time to write a such a descriptive question. – user1007074 Aug 11 '18 at 21:02

1 If you are testing using Postman, make sure you set Origin to * or something for the request header, then Attempt #1 should work. Without this header, Access-Control-Allow-Origin will not be returned in the response header. – tala9999 Jan 25 at 16:59

11 Answers

160

Because you have a very simple CORS policy (Allow all requests from XXX domain), you don't need to make it so complicated. Try doing the following first (A very basic implementation of CORS).

If you haven't already, install the CORS nuget package.

```
Install-Package Microsoft.AspNetCore.Cors
```

In the ConfigureServices method of your startup.cs, add the CORS services.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddCors(); // Make sure you call this previous to AddMvc
    services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
}
```

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory
loggerFactory)
{
    // Make sure you call this before calling app.UseMvc()
    app.UseCors(
        options => options.WithOrigins("http://example.com").AllowAnyMethod()
    );

    app.UseMvc();
}
```

Now give it a go. Policies are for when you want different policies for different actions (e.g. different hosts or different headers). For your simple example you really don't need it. Start with this simple example and tweak as you need to from there.

Further reading : http://dotnetcoretutorials.com/2017/01/03/enabling-cors-asp-net-core/

edited Jun 9 at 0:37

Mauricio Gracia Gutierrez

5,754 3 39 59

answered Jun 6 '17 at 1:16

MindingData

5,953 2 31 43

XMLHttpRequest cannot load andrewgodfroyportfolioapi.azurewebsites.net/api/Authentication. Response to preflight request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'localhost:3000' is therefore not allowed access. The response had HTTP status code 415. – killerrin Jun 6 '17 at 3:53

Unfortunately that didn't work. I added the code as you/the tutorial said (in the order its listed in too), with example.com replaced with localhost:3000 and I get an error – killerrin Jun 6 '17 at 3:56

6 This is unlikely going to work, when you register `app.UseCors` **AFTER** `` app.UseMvc()`. Middlewares are executed in the order they are registered – Tseng Jun 8 '17 at 5:57

16 using app.UseCors before app.UseMvc, in Configure method seems to work. For some reason, the sequence does seem to matter. – MrClan Aug 16 '17 at 5:16

1 I had to enable `options.DisableHttpsRequirement();` in order for any of this to work. It seems with https cors settings were not applying. – Michael Brown Mar 22 '18 at 6:30

```
app.UseCors(builder => builder
    .AllowAnyOrigin()
    .AllowAnyMethod()
    .AllowAnyHeader()
    .AllowCredentials());
app.UseMvc();
```

Main point is that add `app.UseCors` , before `app.UseMvc()` .

Make sure you declare the CORS functionality before MVC so the middleware fires before the MVC pipeline gets control and terminates the request.

After the above method works you can change it configure a specific ORIGIN to accept api calls and avoid leaving your API so open to anyone

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddCors(options => options.AddPolicy("ApiCorsPolicy", builder =>
    {
        builder.WithOrigins("http://localhost:4200").AllowAnyMethod().AllowAnyHeader();
    }));
    services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
}
```

In the configure method tell CORS to use the policy you just created:

```
app.UseCors("ApiCorsPolicy");
app.UseMvc();
```

I just found this compact article on the subject - https://dzone.com/articles/cors-in-net-core-net-core-security-part-vi

edited Jun 9 at 13:55

Mauricio Gracia Gutierrez

5,754 3 39 59

answered Aug 9 '17 at 19:53

Ji Ra

1,697 1 8 12

1 Thank you @Ji Ra – atik sarker May 1 '18 at 15:31

3 Just to mention this, in contrast to `Configure()` the order is not really important here within `ConfigureServices()` – B12Toaster May 1 '18 at 19:32

I used the link in the Further Reader and those steps resolved this error. I wasn't sure where these changes should be placed (I thought the API). The Link confirmed that they should be placed in the API. Thanks for the help. I was totally spinning my wheels with this error. – Richard May 14 '18 at 12:37

22

I created my own middleware class that worked for me, i think there is something wrong with .net core middleware class

```
public class CorsMiddleware
{
    private readonly RequestDelegate _next;

    public CorsMiddleware(RequestDelegate next)
    {
        _next = next;
    }

    public Task Invoke(HttpContext httpContext)
    {
        httpContext.Response.Headers.Add("Access-Control-Allow-Origin", "*");
        httpContext.Response.Headers.Add("Access-Control-Allow-Credentials", "true");
        httpContext.Response.Headers.Add("Access-Control-Allow-Headers", "Content-Type,
X-CSRF-Token, X-Requested-With, Accept, Accept-Version, Content-Length, Content-MD5,
Date, X-Api-Version, X-File-Name");
        httpContext.Response.Headers.Add("Access-Control-Allow-Methods",
"POST,GET,PUT,PATCH,DELETE,OPTIONS");
        return _next(httpContext);
    }
}

// Extension method used to add the middleware to the HTTP request pipeline.
public static class CorsMiddlewareExtensions
{
    public static IApplicationBuilder UseCorsMiddleware(this IApplicationBuilder
builder)
    {
        return builder.UseMiddleware<CorsMiddleware>();
```

```
app.UseCorsMiddleware();
```

answered Aug 23 '17 at 15:55

user8266077
239 2 2

Very elegant way of running Access-Control-Allow-Origin. – Artur Poniedziałek Aug 25 '17 at 12:52

1 thx very elegant. and yes, there is something wrong. – lolol Jan 26 '18 at 21:49

This works on WebAPI and MVC and has no dependencies, Thank you! – Joe May 24 '18 at 2:14

I was skeptical about this also, but it worked for me. I tried basically every other method to accomplish this that I could find on the internet, but no matter what the server would not respond with the access headers. This worked great. I'm running aspnetcore 2.1. – Jordan Jul 19 '18 at 13:43

You should return cors headers only if client send header "Origin" in request. In original CospMiddleware it looks like this: `if (!context.Request.Headers.ContainsKey(CorsConstants.Origin)) return this._next(context);` – Andrew Prigorshnev Jun 27 at 12:43

14

In my case only `get` request works well according to MindingData's answer. For other types of request you need to write:

```
app.UseCors(corsPolicyBuilder =>
   corsPolicyBuilder.WithOrigins("http://localhost:3000")
  .AllowAnyMethod()
  .AllowAnyHeader()
);
```

Don't forget to add `.AllowAnyHeader()`

answered Oct 14 '17 at 10:06

Towhid
1,278 3 30 47

Agree with Towhid that AllowAnyHeader() is needed. It let server receives OPTIONS request if HEADER's request is missing something. – Rivon Oct 18

8

To expand on user8266077's answer, I found that I still needed to supply OPTIONS response for preflight requests in .NET Core 2.1-preview for my use case:

```
// https://stackoverflow.com/a/45844400
public class CorsMiddleware
{
  private readonly RequestDelegate _next;

  public CorsMiddleware(RequestDelegate next)
  {
    _next = next;
  }

  public async Task Invoke(HttpContext context)
  {
    context.Response.Headers.Add("Access-Control-Allow-Origin", "*");
    context.Response.Headers.Add("Access-Control-Allow-Credentials", "true");
    // Added "Accept-Encoding" to this list
    context.Response.Headers.Add("Access-Control-Allow-Headers", "Content-Type, X-CSRF-
Token, X-Requested-With, Accept, Accept-Version, Accept-Encoding, Content-Length,
Content-MD5, Date, X-Api-Version, X-File-Name");
    context.Response.Headers.Add("Access-Control-Allow-Methods",
"POST,GET,PUT,PATCH,DELETE,OPTIONS");
    // New Code Starts here
    if (context.Request.Method == "OPTIONS")
    {
      context.Response.StatusCode = (int)HttpStatusCode.OK;
      await context.Response.WriteAsync(string.Empty);
    }
    // New Code Ends here

    await _next(context);
  }
}
```

and then enabled the middleware like so in Startup.cs

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
```

answered Mar 27 '18 at 18:05

ckr
81 2 2

1 I recommend adding middleware that way: `app.Use<CorsMiddleware>();` – Albert221 Sep 24 '18 at 13:38

You can replace those 2 ligne : context.Response.StatusCode = (int)HttpStatusCode.OK; await context.Response.WriteAsync(string.Empty); with a simple : return; – Hayha Oct 18 '18 at 22:27

To expand on your expansion of @user8266077's answer: Beware that if the request for some other reason fails, this middleware will throw an exception and the headers will not be set. Meaning that in frontend, it will still look like a CORS issue even though it's something totally different. I bypassed this by catching any exceptions in `await _next(context)` and setting the status code and response manually if this happens. I also had to add "authorization" to Access-Control-Allow-Headers for the preflight request to work when making requests from react that requires authorization. – Adam Jan 5 at 0:03

5

None of the above procedures helped and I then read article which solved the issue.

Below is the code.

```
public void ConfigureServices(IServiceCollection services)
{
    // Add service and create Policy with options
    services.AddCors(options =>
    {
        options.AddPolicy("CorsPolicy",
            builder => builder.AllowAnyOrigin()
            .AllowAnyMethod()
            .AllowAnyHeader()
            .AllowCredentials() );
    });


    services.AddMvc();
}
```

and

```
// global policy - assign here or on each controller
app.UseCors("CorsPolicy");
```

and on the top of my actionmethod

```
[EnableCors("CorsPolicy")]
```

answered Dec 17 '18 at 14:04

Sainath
463 2 7 19

1 Very very very very thanks. :) :) :) It worked for me. – namco Mar 15 at 10:40

4

try adding `jQuery.support.cors = true;` before the Ajax call

It could also be that the data your sending to the API is wonky,

try adding the following JSON function

```
    var JSON = JSON || {};

// implement JSON.stringify serialization
JSON.stringify = JSON.stringify || function (obj) {

    var t = typeof (obj);
    if (t != "object" || obj === null) {

        // simple data type
        if (t == "string") obj = '"' + obj + '"';
        return String(obj);

    }
    else {
```

```
                if (t == "string") v = '"' + v + '"';
                else if (t == "object" && v !== null) v = JSON.stringify(v);

                json.push((arr ? "" : '"' + n + '":') + String(v));
            }

            return (arr ? "[" : "{") + String(json) + (arr ? "]" : "}");
        }
    };

    // implement JSON.parse de-serialization
    JSON.parse = JSON.parse || function (str) {
        if (str === "") str = '""';
        eval("var p=" + str + ";");
        return p;
    };
```

then in your data: object change it to

```
data: JSON.stringify({
    username: username,
    password: password
}),
```

edited Sep 25 '17 at 10:39 — Hossein Narimani Rad — 21.9k 12 68 98

answered Jun 6 '17 at 6:46 — Riaan Saayman — 66 2

Thanks for your help. Definitely utilized a portion of the answer to figure out the solution in the end after combining everyones answers – killerrin Jun 8 '17 at 1:27

0

Based on your comment in MindingData's answer, it has nothing to do with your CORS, it's working fine.

Your Controller action is returning the wrong data. HttpCode 415 means, "Unsupported Media type". This happens when you either pass the wrong format to the controller (i.e. XML to a controller which only accepts json) or when you return a wrong type (return Xml in a

Tseng
37.3k 5 105 138

Thanks for your help. Tried a new solution and played with json being sent out and it worked after I stringified it and got it to work – killerrin Jun 8 '17 at 1:31

2

I think if you use your own **CORS** middleware you need to make sure it is really **CORS** request by checking *origin* header.

```
 public class CorsMiddleware
{
    private readonly RequestDelegate _next;
    private readonly IMemoryCache _cache;
    private readonly ILogger<CorsMiddleware> _logger;

    public CorsMiddleware(RequestDelegate next, IMemoryCache cache,
ILogger<CorsMiddleware> logger)
    {
        _next = next;
        _cache = cache;
        _logger = logger;
    }
    public async Task InvokeAsync(HttpContext context, IAdministrationApi adminApi)
    {
        if (context.Request.Headers.ContainsKey(CorsConstants.Origin) ||
context.Request.Headers.ContainsKey("origin"))
        {
            if (!context.Request.Headers.TryGetValue(CorsConstants.Origin, out var
origin))
            {
                context.Request.Headers.TryGetValue("origin", out origin);
            }

            bool isAllowed;
            // Getting origin from DB to check with one from request and save it in
cache
            var result = _cache.GetOrCreateAsync(origin, async cacheEntry => await
adminApi.DoesExistAsync(origin));
```

```
                    CorsConstants.AccessControlAllowHeaders,
                    $"{HeaderNames.Authorization}, {HeaderNames.ContentType},
{HeaderNames.AcceptLanguage}, {HeaderNames.Accept}");
                context.Response.Headers.Add(CorsConstants.AccessControlAllowMethods,
"POST, GET, PUT, PATCH, DELETE, OPTIONS");

                if (context.Request.Method == "OPTIONS")
                {
                    _logger.LogInformation("CORS with origin {Origin} was handled
successfully", origin);
                    context.Response.StatusCode = (int)HttpStatusCode.NoContent;
                    return;
                }

                await _next(context);
            }
            else
            {
                if (context.Request.Method == "OPTIONS")
                {
                    _logger.LogInformation("Preflight CORS request with origin {Origin}
was declined", origin);
                    context.Response.StatusCode = (int)HttpStatusCode.NoContent;
                    return;
                }

                _logger.LogInformation("Simple CORS request with origin {Origin} was
declined", origin);
                context.Response.StatusCode = (int)HttpStatusCode.Forbidden;
                return;
            }
        }

        await _next(context);
    }
```

answered Mar 6 at 6:04

Riddik

437 3 13

kjbetz
81 4

0

I got MindingData's answer above to work, but I had to use Microsoft.AspNet.Cors instead of Microsoft.AspNetCore.Cors. I am using .NetCore Web Application API project in Visual Studio 2019

answered Jun 12 at 15:30

carl
23 4