Enabling CORS In ASP.NET Core API Applications

Gowtham K Updated date Feb 20 2019 107.8k 7 8

What is CORS?

Cross-Origin Resource Sharing (CORS) manages the cross-origin requests. Unlike same-origin policy, CORS allows making a request from one origin to another. CORS allows the servers to specify who can access the resource on the server from outside.

The origin is made up of three parts - the protocol, host, and the port number.

This is in continuation of my last article (create RESTful API using ASP.NET Core with Entity Framework Core) so I highly recommend you to go through my previous article for the basic set up of an ASP.NET Core application.

Cross Domain call

Before enabling the CORS, let's see how the cross-domain call is restricted. Let's create an ASP.NET Core web application.

Step1

Step 2

Select web application (Model-View-Controller) template, as shown in the below figure,

Step 3

Click OK. This will create a web application with a default template.

Step 4

Go to the Index.cshtml page and add the below code and run the application.

```
<script>
   $.ajax({
      url: "https://localhost:44348/api/Libraries/GetAllAuthor",
      success: function (result) {
          console.log(result);
    }
    })
</script>
```

From the above code, you can notice the AJAX call I made to access the API which is not from the same origin. This is from the ASP.NET Core API application which is created in my last article.

Testing the API in the Postman tool.

In the browser console, you will get an error message as shown in the below figure.

Now it's time to Enable CORS in our API application so that we can access it from a different origin.

Enable CORS in ASP.NET Core API Application

Enabling CORS Globally

Open the ASP.NET Core API application which we created in my last article.

Go to Startup.cs file and add the below code in Configure method, which will inject CORS into a container.

```
01. app.UseCors(options => options.AllowAnyOrigin());
```

Add the below code in ConfigureServices method,

```
01. services.AddCors(c =>
02. {
03.     c.AddPolicy("AllowOrigin", options => options.AllowAnyOrigin());
04. });
```

The above code tells that the API's can be accessed from any origin globally.

Run the application,

From the above figure you can notice we got a response from the API successfully and the response it printed in browser console was as expected.

Enabling for origin

Go to Startup.cs file and add the below code in Configure method,

```
app.UseCors(options=>options.WithOrigins("https://localhost:44342"));
```

Add the below code in ConfigureServices method

```
services.AddCors(c =>
            {
                c.AddPolicy("AllowOrigin", options => options.WithOrigins("https://localhost:44342"
            });
```

Go to controller and decorate the action with Enable CORS attribute, as given below,

```
// GET: api/Libraries/GetAllAuthor
      [HttpGet]
      [Route("GetAllAuthor")]
      [EnableCors("AllowOrigin")]
      public IActionResult GetAllAuthor()
      {
          IEnumerable<Author> authors = _libraryRepository.GetAllAuthor();
          return Ok(authors);
      }
```

Now this API can be accessed only from the origin https://localhost:44342.

We can also define EnableCors at the controller level so that all the actions under this controller can be accessed from the origin https://localhost:44342

```
  [Route("api/Libraries")]
    [ApiController]
    [EnableCors("AllowOrigin")]
```

```
        private readonly ILibraryRepository<Author> _libraryRepository;


        public LibrariesController(ILibraryRepository<Author> libraryRepository)
        {
            _libraryRepository = libraryRepository;
        }

        // GET: api/Libraries/GetAllAuthor
        [HttpGet]
        [Route("GetAllAuthor")]

        public IActionResult GetAllAuthor()
        {
            IEnumerable<Author> authors = _libraryRepository.GetAllAuthor();
            return Ok(authors);
        }
    }

}
```

Conclusion

We saw how to enable the CORS in ASP.NET Core API applications, will see more about ASP.NET Core in my future articles. I hope you have enjoyed this article. Your valuable feedback, questions, or comments about this article are always welcomed.

Next Recommended Article

Bundling And Minifying In ASP.NET Core Applications

Gowtham K TOP 100

77 4m 4 3

Gowtham K is awarded as MVP by Microsoft for his exceptional contribution in Microsoft technologies under the category "Developer Technologies" for the year 2016, 2017 and 2018. He has more than 5 years of expe... Read more

https://www.c-sharpcorner.com/members/gowtham-k3

8 7

Type your comment here and press Enter Key (Minimum 10 characters)

I have done exactly as described above using my WebAPI2 application using .NET Core 3.1 and I still get this error: Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at http://mypc/myapi/weatherforecast. (Reason: CORS header 'Access-Control-Allow-Origin' missing). I am running my Angular 8 app using http://localhost:4200 and have specified AllowAnyOrigin in the ConfigureServices method of my Startup.cs as well as app.UseCors("mypoilcy") in the Configure method. Any thoughts on what else to check?

Greg Knierim

1888 4 0

Dec 13, 2019

1 1 Reply

Hi, did you find a solution? I?m having same issue :(

Francisco Castro

1879 13 0

Dec 18, 2019

0

Thanks for supporting.

Hao Hoang

1889 3 0

Sep 28, 2019

0 0 Reply

Uds son de lo mejor

Andres Leonardo Cano Muñoz

1879 13 0

Jul 28, 2019

0 0 Reply

1843 49 0

0 0 Reply

Nice Article

Sagar Jaybhay

1340 557 19.6k

Feb 20, 2019

0 1 Reply

Thank you Sagar Jaybhay :)

Gowtham K

77 22.2k 4m

Feb 21, 2019

0