**Quickest Way to Enable CORS for ASP.NET Core Web API application**

Browser security does not allow other domain to make AJAX requests. This is called [same-origin policy](https://en.wikipedia.org/wiki/Same-origin_policy) which prevents malicious attack from other sites. When you need to allow other sites to make cross-origin requests, we can enable [Cross-origin resource sharing](https://en.wikipedia.org/wiki/Cross-origin_resource_sharing) (CORS).

There is a pretty good documentation about enabling CORS for dotnet core application [here](https://docs.microsoft.com/en-us/aspnet/core/security/cors?view=aspnetcore-2.2).

In terms of Web API application, the quickest way is to add UserCors in Configure method in startup.cs file as below.

[cc lang="csharp" tab\_size="4" theme="GeSHi" lines="-1"]

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseCors(builder =>

builder.AllowAnyOrigin()

.AllowAnyMethod()

.AllowAnyHeader());

app.UseMvc();

}

[/cc]

Obviously, the above solution should not be used in production. It is supposed to be a quick test in development environment to make your AJAX requests from another domain work.

In reality, we should have specific origins configured alongside with allowed method or headers. Below is the example of allowing cross-origin requests from the specific domain.

[cc lang="csharp" tab\_size="4" theme="GeSHi" lines="-1"]

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseCors(builder =>

builder.WithOrigins("http://helloworld.com")

.WithMethods("GET", "POST", "PUT", "DELETE")

.AllowAnyHeader());

app.UseMvc();

}

[/cc]

Check out a full ASP.NET API example [here](https://github.com/mydatahack/web-api-example-c-sharp-mysql-actor).