**CORS Issues**

Cross-Origin Resource Sharing (CORS) is a security feature that **allows or restricts** web applications running at **one domain** to make requests for resources from a different domain. In .NET Core API, CORS is configured to handle the below scenarios.

1. Domains.
2. HTTP methods.
3. HTTP headers.

**How to enable CORS programmatically**

In a .NET Core API, CORS is implemented using ***middleware***. You typically configure CORS in the Startup.cs file. You need to specify which domains are allowed to access your API. This is done by ***configuring the CORS policy***. A CORS policy defines which ***domains, HTTP methods, headers***, and other options are permitted. You can define ***one or more*** policies with different configurations.

services.AddCors(options =>  
{  
 options.AddPolicy("Policy1", builder =>  
 {  
 builder.WithOrigins("https://facebook.com")  
 .WithMethods("GET", "POST")  
 .WithHeaders("Content-Type");  
 });  
  
 options.AddPolicy("Policy2", builder =>  
 {  
 builder.WithOrigins("https://google.com")  
 .WithMethods("GET", "PUT", "DELETE")  
 .WithHeaders("Authorization", "Content-Type");  
 });  
});

You need to register the CORS middleware in the Configure method of Startup.cs .

app.UseCors("AllowSpecificOrigin");

Run a NuGet Package restore to generate this file.

NuGet Package Restore restores all of a project's dependencies that are listed in either a project file or a packages.config file. You can restore packages manually with nuget restore**, dotnet restore**, msbuild -t:restore, or through Visual Studio.

The dotnet build and dotnet run commands automatically restore packages, and you can configure Visual Studio to restore packages automatically when it builds a project.

**Use Microsoft Store to download installations s.a. Visual Studio Code, Visual Studio 2022.**