[**Web API 2 对 CORS 的支持**](https://www.cnblogs.com/Leo_wl/p/4383069.html)

[**Web API 2 对 CORS 的支持**](http://www.cnblogs.com/drzhong/p/4381904.html)

**CORS概念**

跨域资源共享 (CORS) 是一种万维网联合会 (W3C) 规范（通常被认为是 HTML5 的一部分），它可让 JavaScript 克服由浏览器施加的同域策略安全限制。 所谓同域策略，就是 JavaScript 只能对包含网页的同一个域进行 AJAX 回调（其中，“域”就是主机名、协议和端口号的组合）。 例如，http://foo.com 中某个网页上的 JavaScript 无法对 http://bar.com（或 http://www.foo.com、https://foo.com 或 http://foo.com:999 等）进行 AJAX 调用。

CORS 可让服务器指明允许哪些域对它们进行调用，从而放宽这种限制。 CORS 是由浏览器强制执行的，并且必须在服务器上实现，而最新版本的 ASP.NET Web API 2 全面支持 CORS。 通过 Web API 2，您可以对策略进行配置以允许不同域的 JavaScript 客户端访问您的 API。

**CORS 基本信息**

由于 Web API 完全按照该规范来实现，因此，为了使用 Web API 中的新 CORS 功能，详细了解 CORS 本身将大有帮助。 这些详细内容现在看起来可能都是理论之谈，但对于以后了解 Web API 中的可用设置来说将十分有用：在您调试 CORS 时，这些内容有助于您更快速地解决问题。

CORS 的一般机制是，当 JavaScript 尝试进行跨域 AJAX 调用时，浏览器会通过在 HTTP 请求中发送标头（如“Origin”）来“询问”服务器是否允许进行这样的调用。 服务器通过在响应中返回 HTTP 标头（如“Access-Control-Allow-Origin”）指明允许的操作。 这种权限检查将针对客户端调用的每个不同 URL 进行，这就意味着不同的 URL 可以具有不同权限。

除域之外，CORS 还可以让服务器指明允许使用的 HTTP 方法、客户端可以发送的 HTTP 请求标头、客户端可以读取的 HTTP 响应标头以及是否允许浏览器自动发送或接收凭据（Cookie 或授权标头）。 其他请求和响应标头指明允许使用其中的哪些功能。 **图 1** 总结了这些标头（请注意，一些功能没有在响应中发送的标头，仅有响应）。

**图 1 CORS HTTP 标头**

|  |  |  |
| --- | --- | --- |
| 权限/功能 | 请求标头 | 响应标头 |
| 域 | 域 | Access-Control-Allow-Origin |
| HTTP 方法 | Access-Control-Request-Method | Access-Control-Allow-Method |
| 请求标头 | Access-Control-Request-Headers | Access-Control-Allow-Headers |
| 响应标头 |  | Access-Control-Expose-Headers |
| 凭据 |  | Access-Control-Allow-Credentials |
| 缓存预检响应 |  | Access-Control-Max-Age |

浏览器可通过两种不同的方式向服务器请求这些权限：简单 CORS 请求和预检 CORS 请求。

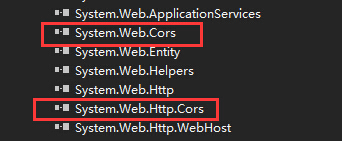
**Web API 2 对 CORS 的支持**

Web API 中对 CORS 的支持是一个完整框架，允许应用程序定义 CORS 请求的权限。 该框架围绕一个策略方案展开，该策略方案可让您指定针对进入应用程序的任何给定请求而允许的 CORS 功能。

首先，为了获取该 CORS 框架，您必须从 Web API 应用程序引用 CORS 库（默认情况下，Visual Studio 2013 中的任何 Web API 模板都不引用这些库）。 该 Web API CORS 框架通过 NuGet 作为 Microsoft.AspNet.WebApi.Cors 程序包提供。 早 nuget中输入

Install-Package Microsoft.AspNet.WebApi.Cors

注意 Web api 2对,net framework的要求必须是4.5以上，安装完上面的package后，会发现引用中多了两个重要的包，如下图所示



接下来，为了表达该策略，Web API 提供了一个名为 EnableCorsAttribute 的自定义属性类。 此类包含允许的域、HTTP 方法、请求标头、响应标头以及是否允许使用凭据等方面的属性（它们对前面所述的 CORS 规范的所有详细信息进行建模）。

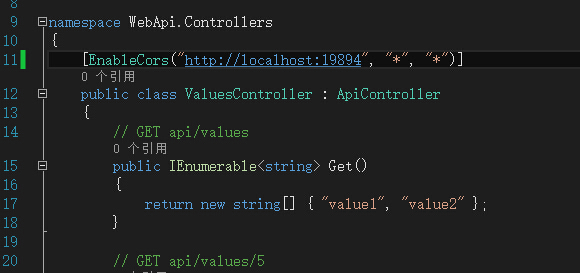
最后，为了让 Web API CORS 框架处理 CORS 请求并发出适当的 CORS 响应标头，该类必须检查进入应用程序的每个请求。 Web API 通过消息处理程序提供用于这种拦截操作的扩展点。 Web API CORS 框架会相应地实现一个名为 CorsMessageHandler 的消息处理程序。 对于 CORS 请求，该处理程序会查询在所调用方法的属性中表达的策略，并发出适当的 CORS 响应标头。

**EnableCorsAttribute**。EnableCorsAttribute 类就是应用程序表达其 CORS 策略的方式。EnableCorsAttribute 类有一个可接受三个或四个参数的重载构造函数。 这些参数（依次）为：

1. 允许域列表
2. 允许请求标头列表
3. 允许 HTTP 方法列表
4. 允许响应标头列表（可选）

还有一个允许使用凭据的属性 (Supports­Credentials) 以及另一个用于指定预检缓存持续时间值的属性 (PreflightMaxAge)。

以vs2013建立的默认api程序为例，建完webapi引用程序后在Controller文件夹下会自动生成两个控制区，一个HomeController和一个ValueController，我们主要看一下ValueControll，这个控制器继承了ApiController，可见是一个webapi，我们在valueController上添加一个全局的 EnableCors属性，以试它支持跨域，如下图所示



请注意，每个构造函数参数都是一个字符串。 通过指定逗号分隔列表，可以表示多个值。 如果您想允许所有域、请求标头或 HTTP 方法，则可以使用“\*”作为值（对于响应标头仍须显式指定）。

除在方法级别应用 EnableCors 属性外，还可以在类级别应用该属性，或将其全局应用于应用程序。 应用该属性的级别会在 Web API 代码中为该级别及下面级别的所有请求配置 CORS。 例如，如果在方法级别应用该策略，则该策略仅应用于该操作的请求，而如果在类级别应用该策略，则该策略将应用于对该控制器的所有请求。 最后，如果全局应用该策略，则该策略将应用于所有请求。

如果在多个位置存在策略，则会使用“最接近的”属性，并忽略其他属性（优先顺序是方法、类、全局）。 如果您已在较高级别上应用策略，但随后想在较低级别上排除某一请求，则可以使用名为 DisableCorsAttribute 的另一个属性类。 此属性实质上是一个没有允许权限的策略。

如果在您不想允许 CORS 的控制器上有其他方法，则有两个选择。 首先，您可在 HTTP 方法列表中显式指定。 或者，您可以保留通配符，但使用 DisableCors 属性来排除 Delete 方法。

**CorsMessageHandler**。必须为 CORS 框架启用 CorsMessageHandler，才能执行其为评估 CORS 策略并发出 CORS 响应标头而拦截请求的工作。 消息处理程序的启用通常是在应用程序的 Web API 配置类中通过调用 EnableCors 扩展方法进行的：

**[复制代码](javascript:void(0);)**

复制代码

public static class WebApiConfig

{

public static void Register(HttpConfiguration config)

{

// Web API 配置和服务

// Web API 路由

config.MapHttpAttributeRoutes();

config.Routes.MapHttpRoute(

name: "DefaultApi",

routeTemplate: "api/{controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

　　　　　　 //启用跨域

config.EnableCors();

}

}

复制代码

**[复制代码](javascript:void(0);)**