<http://www.cnblogs.com/landeanfen/p/5177176.html>

# [C#进阶系列——WebApi 跨域问题解决方案：CORS](http://www.cnblogs.com/landeanfen/p/5177176.html)

**阅读目录**

* [一、跨域问题的由来](http://www.cnblogs.com/landeanfen/p/5177176.html#_label0)
* [二、跨域问题解决原理](http://www.cnblogs.com/landeanfen/p/5177176.html#_label1)
* [三、跨域问题解决细节](http://www.cnblogs.com/landeanfen/p/5177176.html#_label2)
  + [1、场景描述](http://www.cnblogs.com/landeanfen/p/5177176.html#_label2_0)
  + [2、场景测试](http://www.cnblogs.com/landeanfen/p/5177176.html#_label2_1)
* [四、总结](http://www.cnblogs.com/landeanfen/p/5177176.html#_label3)

**正文**

前言：上篇总结了下WebApi的接口测试工具的使用，这篇接着来看看WebAPI的另一个常见问题：跨域问题。本篇主要从实例的角度分享下CORS解决跨域问题一些细节。

WebApi系列文章

* [C#进阶系列——WebApi接口测试工具：WebApiTestClient](http://www.cnblogs.com/landeanfen/p/5210356.html)
* [C#进阶系列——WebApi 跨域问题解决方案：CORS](http://www.cnblogs.com/landeanfen/p/5177176.html)
* [C#进阶系列——WebApi身份认证解决方案：Basic基础认证](http://www.cnblogs.com/landeanfen/p/5287064.html)
* [C#进阶系列——WebApi接口传参不再困惑：传参详解](http://www.cnblogs.com/landeanfen/p/5337072.html)
* [C#进阶系列——WebApi接口返回值不困惑：返回值类型详解](http://www.cnblogs.com/landeanfen/p/5501487.html)
* C#进阶系列——WebApi异常处理解决方案
* C#进阶系列——WebApi区域Area使用小结

[回到顶部](http://www.cnblogs.com/landeanfen/p/5177176.html#_labelTop)

# 一、跨域问题的由来

**同源策略：出于安全考虑，浏览器会限制脚本中发起的跨站请求，浏览器要求JavaScript或Cookie只能访问同域下的内容。**

正是由于这个原因，我们不同项目之间的调用就会被浏览器阻止。比如我们最常见的场景：WebApi作为数据服务层，它是一个单独的项目，我们的MVC项目作为Web的显示层，这个时候我们的MVC里面就需要调用WebApi里面的接口取数据展现在页面上。因为我们的WebApi和MVC是两个不同的项目，所以运行起来之后就存在上面说的跨域的问题。

[回到顶部](http://www.cnblogs.com/landeanfen/p/5177176.html#_labelTop)

# 二、跨域问题解决原理

CORS全称Cross-Origin Resource Sharing，中文全称跨域资源共享。它解决跨域问题的原理是通过向http的请求报文和响应报文里面加入相应的标识告诉浏览器它能访问哪些域名的请求。比如我们向响应报文里面增加这个Access-Control-Allow-Origin:http://localhost:8081，就表示支持http://localhost:8081里面的所有请求访问系统资源。其他更多的应用我们就不一一列举，可以去网上找找。

[回到顶部](http://www.cnblogs.com/landeanfen/p/5177176.html#_labelTop)

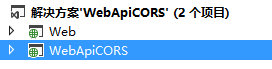
# 三、跨域问题解决细节

 下面我就结合一个简单的实例来说明下如何使用CORS解决WebApi的跨域问题。

[回到顶部](http://www.cnblogs.com/landeanfen/p/5177176.html#_labelTop)

## 1、场景描述

我们新建两个项目，一个WebApi项目（下图中WebApiCORS），一个MVC项目（下图中Web）。WebApi项目负责提供接口服务，MVC项目负责页面呈现。如下：



其中，**Web与WebApiCORS端口号分别为“27239”和“27221”。**Web项目需要从WebApiCORSS项目里面取数据，很显然，两个项目端口不同，所以并不同源，如果使用常规的调用方法肯定存在一个跨域的问题。

简单介绍下测试代码，Web里面有一个HomeController

[复制代码](javascript:void(0);)

　　public class HomeController : Controller

{

// GET: Home

public ActionResult Index()

{

return View();

}

}

[复制代码](javascript:void(0);)

对应的Index.cshtml

[复制代码](javascript:void(0);)

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

<script src="~/Content/jquery-1.9.1.js"></script>

<link href="~/Content/bootstrap/css/bootstrap.css" rel="stylesheet" />

<script src="~/Content/bootstrap/js/bootstrap.js"></script>

<script src="~/Scripts/Home/Index.js"></script>

</head>

<body>

测试结果：<div id="div\_test">

</div>

</body>

</html>

[复制代码](javascript:void(0);)

Index.js文件

[复制代码](javascript:void(0);)

var ApiUrl = "http://localhost:27221/";

$(function () {

$.ajax({

type: "get",

url: ApiUrl + "api/Charging/GetAllChargingData",

data: {},

success: function (data, status) {

if (status == "success") {

$("#div\_test").html(data);

}

},

error: function (e) {

$("#div\_test").html("Error");

},

complete: function () {

}

});

});

[复制代码](javascript:void(0);)

WebApiCORS项目里面有一个测试的WebApi服务ChargingController

[复制代码](javascript:void(0);)

　　public class ChargingController : ApiController

{

/// <summary>

/// 得到所有数据

/// </summary>

/// <returns>返回数据</returns>

[HttpGet]

public string GetAllChargingData()

{

return "Success";

}

}

[复制代码](javascript:void(0);)

配置WebApi的路由规则为通过action调用。WebApiConfig.cs文件

[复制代码](javascript:void(0);)

　　 public static class WebApiConfig

{

public static void Register(HttpConfiguration config)

{

// Web API 路由

config.MapHttpAttributeRoutes();

config.Routes.MapHttpRoute(

name: "DefaultApi",

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

defaults: new { id = RouteParameter.Optional }

);

}

}

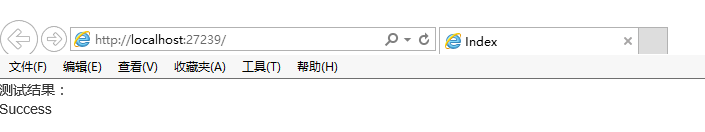
[复制代码](javascript:void(0);)

[回到顶部](http://www.cnblogs.com/landeanfen/p/5177176.html#_labelTop)

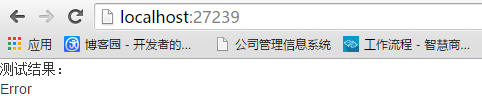
## 2、场景测试

### 1）我们不做任何的处理，直接将两个项目运行起来。看效果如何

 IE浏览器：



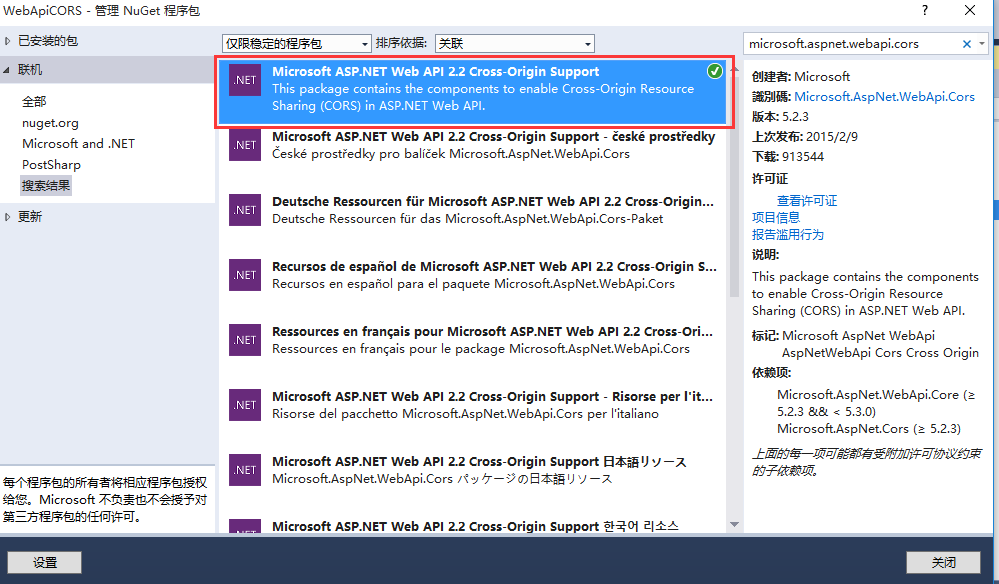
谷歌浏览器：



这个结果另博主也很吃惊，不做任何跨域处理，IE10、IE11竟然可以直接请求数据成功，而同样的代码IE8、IE9、谷歌浏览器却不能跨域访问。此原因有待查找，应该是微软动了什么手脚。

### 2）使用CORS跨域

首先介绍下CORS如何使用，在WebApiCORS项目上面使用Nuget搜索“microsoft.aspnet.webapi.cors”，安装第一个



然后在App\_Start文件夹下面的WebApiConfig.cs文件夹配置跨域

[复制代码](javascript:void(0);)

public static class WebApiConfig

{

public static void Register(HttpConfiguration config)

{

//跨域配置

config.EnableCors(new EnableCorsAttribute("\*", "\*", "\*"));

// Web API 路由

config.MapHttpAttributeRoutes();

config.Routes.MapHttpRoute(

name: "DefaultApi",

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

defaults: new { id = RouteParameter.Optional }

);

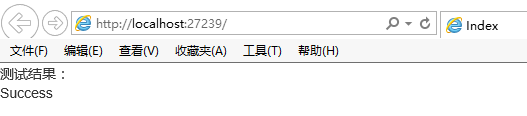
}

}

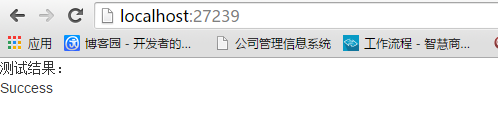
[复制代码](javascript:void(0);)

我们暂定三个“\*”号，当然，在项目中使用的时候一般需要指定对哪个域名可以跨域、跨域的操作有哪些等等。这个在下面介绍。

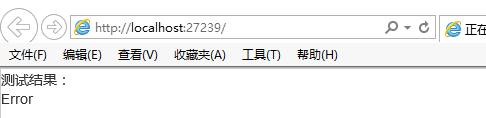
IE10、IE11



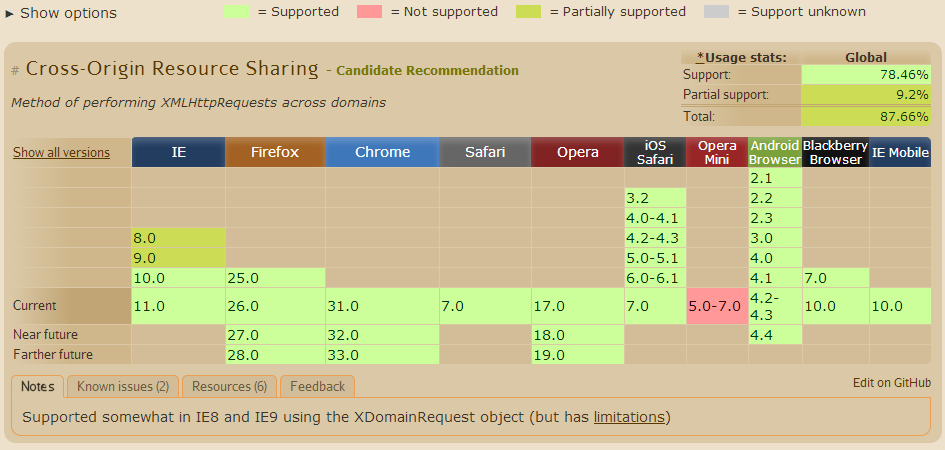
谷歌浏览器



IE8、IE9



这个时候又有新问题了，怎么回事呢？我都已经设置跨域了呀，怎么IE8、9还是不行呢？这个时候就有必要说说CORS的浏览器支持问题了。网上到处都能搜到这张图：



上图描述了CORS的浏览器支持情况，可以看到IE8、9是部分支持的。网上说的解决方案都是Internet Explorer 8 、9使用 XDomainRequest 对象实现CORS。是不是有这么复杂？于是博主各种百度寻找解决方案。最后发现在调用处指定 jQuery.support.cors = true; 这一句就能解决IE8、9的问题了。具体是在Index.js里面

[复制代码](javascript:void(0);)

jQuery.support.cors = true;

var ApiUrl = "http://localhost:27221/";

$(function () {

$.ajax({

type: "get",

url: ApiUrl + "api/Charging/GetAllChargingData",

data: {},

success: function (data, status) {

if (status == "success") {

$("#div\_test").html(data);

}

},

error: function (e) {

$("#div\_test").html("Error");

},

complete: function () {

}

});

});

[复制代码](javascript:void(0);)

这句话的意思就是指定浏览器支持跨域。原来IE9以上版本的浏览器、谷歌、火狐等都默认支持跨域，而IE8、9却默认不支持跨域，需要我们指定一下。你可以在你的浏览器里面打印jQuery.support.cors看看。这样设置之后是否能解决问题呢？我们来看效果：



问题完美解决。至于网上说的CORS对IE8、9的解决方案XDomainRequest是怎么回事，有待实例验证。

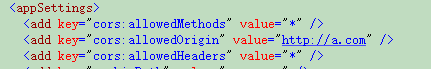
### 3）CORS的具体参数设置。

上文我们使用

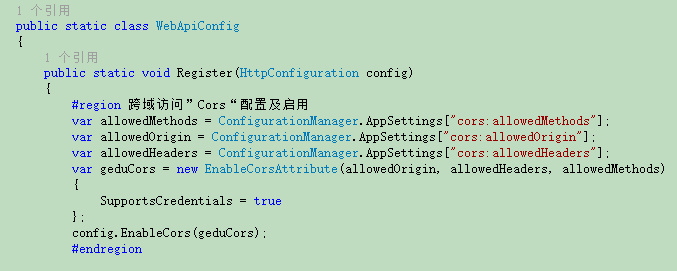
config.EnableCors(new EnableCorsAttribute("\*", "\*", "\*"));

这一句解决了跨域问题，上面说了，这种\*号是不安全的。因为它表示只要别人知道了你的请求url，任何请求都可以访问到你的资源。这是相当危险的。所以需要我们做一些配置，限制访问权限。比如我们比较常见的做法如下：

 配置方法一、在Web.Config里面（PS：这两张图源自：[http://www.cnblogs.com/moretry/p/4154479.html](http://www.cnblogs.com/moretry/p/4154479.html" \t "_blank)）



然后在WebApiConfig.cs文件的Register方法里面



配置方法二、如果你只想对某一些api做跨域，可以直接在API的类上面使用特性标注即可。

[复制代码](javascript:void(0);)

　　[EnableCors(origins: "http://localhost:8081/", headers: "\*", methods: "GET,POST,PUT,DELETE")]

public class ChargingController : ApiController

{

/// <summary>

/// 得到所有数据

/// </summary>

/// <returns>返回数据</returns>

[HttpGet]

public string GetAllChargingData()

{

return "Success";

}

}

[复制代码](javascript:void(0);)

[回到顶部](http://www.cnblogs.com/landeanfen/p/5177176.html#_labelTop)

# 四、总结

以上就是一个简单的CORS解决WebApi跨域问题的实例，由于博主使用WebApi的时间并不长，所以很多理论观点未必成熟，如果有说的不对的，欢迎指出。博主在此多谢啦。

**2、config方式实现CORS**

在Web.config的system.webServer配置节下增加配置，这种方式的好处是简单，只要在这里加了这个配置，那么所有的api都可以按同一种规则支持跨域请求

**[html]** [view plain](http://blog.csdn.net/starfd/article/details/45307659) [copy](http://blog.csdn.net/starfd/article/details/45307659)

1. <httpProtocol>
2. <customHeaders>
3. <add name="Access-Control-Allow-Origin" value="\*" />
4. <add name="Access-Control-Allow-Headers" value="\*" />
5. <add name="Access-Control-Allow-Methods" value="GET, POST, PUT, DELETE" />
6. </customHeaders>
7. </httpProtocol>

如果是HTTP Basic Access authentication，似乎还需要个

<add name="Access-Control-Allow-Credentials" value="true" />，

此处还未验证

<configuration>

<system.webServer>

<httpProtocol>  
    <customHeaders>  
        <add name="Access-Control-Allow-Origin" value="\*" />  
        <add name="Access-Control-Allow-Methods" value="\*" />

<add name="Access-Control-Allow-Headers"

value="Origin, X-Requested-With, Content-Type, Accept, Authorization"/>  
      </customHeaders>

</httpProtocol>

s</system.webServer>

</configuration>