Moving forward with my mobile app development learning process, I've found a new obstacle: Cross-origin Request Sharing or CORS.

I am using a combination of AngularJS + jQuery Mobile (Cordova phone client) and ASP.NET Web API (backend). My issue is that I have not been able to complete a POST request (or any other type of request) to an API controller.

My AngularJS controller uses the $http.post() service method to call the Web API controller. However, Chrome debugger says that the call failed in an OPTIONS request (possibly the CORS preflight request).

I have implemented the CORS action selector from the following post: [Enabling CORS in Web API Project](http://code.msdn.microsoft.com/windowsdesktop/CORS-support-in-ASPNET-Web-01e9980a). Even tough I can call the api method from Fiddler, AngularJS keeps failing on the OPTIONS preflight request.

Is there anything I should be aware of about AngularJS and cross-domain calls? Any possible solution to my predicament?

Thanks.

You can skip the preflight option request by using content-type : application/x-www-form-urlencoded.

**AngularJS:**

var user = {

ID: 1,

Name: 'test'

};

$http({

url: "api.localhost/api/users/addUser",

method: "POST",

data: $.param(user),

headers: { 'Content-Type': 'application/x-www-form-urlencoded' },

withCredentials: true,

}).success(function (data, status, headers, config) {

console.log(data);

})

**Web api:**

[HttpPost]

public string AddUser(User user)

{

// Do something

return user.Name + " added";

}

**Web.config**

<system.webServer>

<validation validateIntegratedModeConfiguration="false" />

<handlers>

<remove name="ExtensionlessUrlHandler-ISAPI-4.0\_32bit" />

<remove name="ExtensionlessUrlHandler-ISAPI-4.0\_64bit" />

<remove name="ExtensionlessUrlHandler-Integrated-4.0" />

<remove name="OPTIONSVerbHandler" />

<remove name="TRACEVerbHandler" />

<add name="ExtensionlessUrlHandler-ISAPI-4.0\_32bit" path="\*." verb="GET,HEAD,POST,DEBUG,PUT,DELETE,PATCH,OPTIONS" modules="IsapiModule" scriptProcessor="%windir%\Microsoft.NET\Framework\v4.0.30319\aspnet\_isapi.dll" preCondition="classicMode,runtimeVersionv4.0,bitness32" responseBufferLimit="0" />

<add name="ExtensionlessUrlHandler-ISAPI-4.0\_64bit" path="\*." verb="GET,HEAD,POST,DEBUG,PUT,DELETE,PATCH,OPTIONS" modules="IsapiModule" scriptProcessor="%windir%\Microsoft.NET\Framework64\v4.0.30319\aspnet\_isapi.dll" preCondition="classicMode,runtimeVersionv4.0,bitness64" responseBufferLimit="0" />

<add name="ExtensionlessUrlHandler-Integrated-4.0" path="\*." verb="\*" type="System.Web.Handlers.TransferRequestHandler" preCondition="integratedMode,runtimeVersionv4.0" />

</handlers>

<httpProtocol>

<customHeaders>

<add name="Access-Control-Allow-Origin" value="http://localhost" />

<add name="Access-Control-Allow-Headers" value="Origin, X-Requested-With, Content-Type, Accept, Cache-Control" />

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

<add name="Access-Control-Allow-Methods" value="GET, POST, PUT, DELETE, OPTIONS" />

</customHeaders>

</httpProtocol>

</system.webServer>

While stumbling onto this issue with AngularJS 1.3 with Microsoft Web API 2 I found a simple solution to the CORS configuration issue.

* First from Nuget intall - Microsoft WebApi Cors

Install-Package Microsoft.AspNet.WebApi.Cors

* Then in your WebApiConfig.cs file:
* var cors = new System.Web.Http.Cors.EnableCorsAttribute("www.my-angular-web-site.com", "\*", "\*");

config.EnableCors(cors);

You can also enable CORS everywhere with a \* instead of your web site but that defeats the purpose of CORS and opens up security holes - but you can do it for testing.

The WebApi.Cors assembly also lets you enable cors controller by controller or with other more granular details. Other details can be found [here](http://www.asp.net/web-api/overview/security/enabling-cross-origin-requests-in-web-api) on the Web API site.

Check out Thinktecture Cors objects, with those (nugettable) you can get full CORS support in WebApi without any code of your own.

Even with a correct CORS implementation I've had a really strange issue with IIS 7 that was solved by enabling all verbs for WebDav (yes, WebDav - don't ask me why I just followed instructions on a blog post :-D).

Do this:

* Open IIS manager, go to your application's Handler Mapping.
* Double click the WebDav handler
* Click "Request Restriction"
* On the "Verbs" tab, select "All verbs".

Again, no idea why WebApi uses WebDav, but this solved problems with POST and DELETE that wouldn't work in spite of a correct CORS implementation.

PHP

header('Access-Control-Allow-Origin: \*');

header('Access-Control-Allow-Methods: GET, POST');

private function set\_headers() {

header("HTTP/1.1 ".$this->\_code." ".$this->get\_status\_message());

header("Content-Type:".$this->\_content\_type);

header("Access-Control-Allow-Origin: \*");

}