Cross-Origin Request Headers(CORS) with PHP headers

Asked 12 years, 11 months ago Modified 7 months ago Viewed 639k times Part of PHP Collective

263

I have a simple PHP script that I am attempting a cross-domain CORS request:

```
<?php
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Headers: *");
...
```

Yet I still get the error:

> Request header field `X-Requested-With` is not allowed by `Access-Control-Allow-Headers`

Anything I'm missing?

PHP php javascript xmlhttprequest cors

Share
Improve this question
Follow

edited Dec 16, 2019 at 13:24

Machavity ♦
31.6k ● 27 ● 95 ● 105

asked Jan 3, 2012 at 22:03

user1022241

11 Answers

Sorted by: Highest score (default)

428

Handling CORS requests properly is a tad more involved. Here is a function that will respond more fully (and properly).

```
/**
 *  An example CORS-compliant method.  It will allow any GET, POST, or OPTIONS
requests from any
 *  origin.
 *
 *  In a production environment, you probably want to be more restrictive, but
this gives you
 *  the general idea of what is involved.  For the nitty-gritty low-down, read:
 *
 *  - https://developer.mozilla.org/en/HTTP_access_control
 *  - https://fetch.spec.whatwg.org/#http-cors-protocol
 *
```

```
 */
function cors() {

    // Allow from any origin
    if (isset($_SERVER['HTTP_ORIGIN'])) {
        // Decide if the origin in $_SERVER['HTTP_ORIGIN'] is one
        // you want to allow, and if so:
        header("Access-Control-Allow-Origin: {$_SERVER['HTTP_ORIGIN']}");
        header('Access-Control-Allow-Credentials: true');
        header('Access-Control-Max-Age: 86400');    // cache for 1 day
    }

    // Access-Control headers are received during OPTIONS requests
    if ($_SERVER['REQUEST_METHOD'] == 'OPTIONS') {

        if (isset($_SERVER['HTTP_ACCESS_CONTROL_REQUEST_METHOD']))
            // may also be using PUT, PATCH, HEAD etc
            header("Access-Control-Allow-Methods: GET, POST, OPTIONS");

        if (isset($_SERVER['HTTP_ACCESS_CONTROL_REQUEST_HEADERS']))
            header("Access-Control-Allow-Headers:
{$_SERVER['HTTP_ACCESS_CONTROL_REQUEST_HEADERS']}");

        exit(0);
    }

    echo "You have CORS!";
}
```

Security Notes

> **Check the HTTP_ORIGIN header against a list of approved origins.**

If the origin isn't approved, then you should deny the request.

Please read the spec.

TL;DR

When a browser wants to execute a cross-site request it first confirms that this is okay with a "pre-flight" request to the URL. By allowing CORS you are telling the browser that responses from this URL can be shared with other domains.

CORS does not protect your server. CORS attempts to protect your users by telling browsers what the restrictions should be on sharing responses with other domains. Normally this kind of sharing is utterly forbidden, so CORS is a way to poke a hole in the browser's normal security policy. These holes should be as small as possible, so always check the HTTP_ORIGIN against some kind of internal list.

There are some dangers here, especially if the data the URL serves up is normally protected. You are effectively allowing browser content that originated on some other server to read (and possibly manipulate) data on your server.

If you are going to use CORS, please read the protocol carefully (it is quite small) and try to understand what you're doing. A reference URL is given in the code sample for that purpose.

Header security

It has been observed that the HTTP_ORIGIN header is insecure, and that is true. In fact, all HTTP headers are insecure to varying meanings of the term. Unless a header includes a verifiable signature/hmac, or the whole conversation is authenticated via TLS, headers are just "something the browser has told me".

In this case, the browser is saying "an object from domain X wants to get a response from this URL. Is that okay?" The point of CORS is to be able to answer, "yes I'll allow that".

Share

Improve this answer

Follow

edited Sep 3, 2021 at 3:12

Matt

74.3k ● 10 ● 168 ● 166

answered Mar 26, 2012 at 3:05

slashingweapon

11.3k ● 4 ● 32 ● 52

49 Note that sending the HTTP Origin value back as the allowed origin will allow anyone to send requests to you with cookies, thus potentially stealing a session from a user who logged into your site then viewed an attacker's page. You either want to send '*' (which will disallow cookies thus preventing session stealing) or the specific domains for which you want the site to work. – Jules Jul 7, 2012 at 19:03

1 Agreed. In practice you probably wouldn't allow just any old domain to use your CORS service, you would restrict it to some set that you decided to trust. – slashingweapon Aug 31, 2012 at 17:05

1 The only that's truly work!.. Just change Access-Control-Allow-Origin: * TO Access-Control-Allow-Origin: {$_SERVER['HTTP_ORIGIN']} – Renan Franca Sep 17, 2015 at 18:06

1 By unconditionally allowing any origin with `ACAC: true`, you're essentially throwing the Same-Origin Policy out the window. This answer is terrible advice from a security point of view, and it should be downvoted to oblivion. – jub0bs Jun 16, 2020 at 9:52

2 It is true that `$_SERVER['HTTP_ORIGIN]` is not "secure" in the sense that your app has no way of verifying the true origin of the request. However, it is the **browser's** job to protect this header. Your app is not trying to prevent people from various orgs from using it. Rather, your app is confirming to the browser that cross-site requests from certain domains are acceptable at this URL. – slashingweapon Oct 6, 2020 at 22:00

I got the same error, and fixed it with the following PHP in my back-end script:

121

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

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

header("Access-Control-Allow-Headers: X-Requested-With");
```

Share

Improve this answer

Follow

edited May 30, 2018 at 20:43

Community Bot

1 • 1

answered Sep 4, 2014 at 9:06

Fiach Reid

7,009 • 3 • 34 • 35

should these three lines added in top of PHP file ? Like <?php header('Access-Control-Allow-Origin: *'); header('Access-Control-Allow-Methods: GET, POST'); header("Access-Control-Allow-Headers: X-Requested-With"); – DistributedAI Jun 21, 2022 at 18:40

I think you need to also allow OPTIONS in the Access-Control-Allow-Methods – Svetoslav Marinov Mar 5 at 10:52

80

`Access-Control-Allow-Headers` does not allow `*` as accepted value, see the Mozilla Documentation here.

Instead of the asterisk, you should send the accepted headers (first `X-Requested-With` as the error says).

Update:

`*` is now accepted is `Access-Control-Allow-Headers`.

According to MDN Web Docs 2021:

> The value `*` only counts as a special wildcard value for requests without credentials (requests without HTTP cookies or HTTP authentication information). In requests with credentials, it is treated as the literal header name `*` without special semantics. Note that the Authorization header can't be wildcarded and always needs to be listed explicitly.

Share

Improve this answer

Follow

edited Jun 1, 2021 at 7:40

answered Jan 3, 2012 at 22:10

KARASZI István

31.4k • 9 • 106 • 131

4 As of 2021, it looks like `*` is now accepted as per the MDN docs. – undefined Jan 21, 2021 at 21:41

52

Many description internet-wide don't mention that specifying `Access-Control-Allow-Origin` is not enough. Here is a complete example that works for me:

```
<?php
    if ($_SERVER['REQUEST_METHOD'] === 'OPTIONS') {
        header('Access-Control-Allow-Origin: *');
        header('Access-Control-Allow-Methods: POST, GET, DELETE, PUT, PATCH,
OPTIONS');
        header('Access-Control-Allow-Headers: token, Content-Type');
        header('Access-Control-Max-Age: 1728000');
        header('Content-Length: 0');
        header('Content-Type: text/plain');
        die();
    }

    header('Access-Control-Allow-Origin: *');
    header('Content-Type: application/json');

    $ret = [
        'result' => 'OK',
    ];
    print json_encode($ret);
```

Share Improve this answer Follow

answered Jul 26, 2017 at 14:23

Csongor Halmai
3,845 ● 34 ● 35

5 Please explain why it isn't enough and what minimal example *is* enough. – halfpastfour.am Aug 20, 2018 at 19:09

Unfortunately, I don't remember exactly and I have no time now to investigate it again but, as much as I remember, there were some basic assumptions from the webserver's/browser's side which made it not working. This was the minimal code that worked for me. – Csongor Halmai Aug 26, 2018 at 6:26

if already sent in virtul host of apache ..then only this code work ..if ($_SERVER['REQUEST_METHOD'] === 'OPTIONS') { die(); } – ashutosh Jun 1, 2021 at 7:17

Basically what it says here is that it *is* enough, just not if the request method is 'options'. – jberculo Jun 2, 2021 at 15:40

33

If you want to create a CORS service from PHP, you can use this code as the first step in your file that handles the requests:

```
// Allow from any origin
if(isset($_SERVER["HTTP_ORIGIN"]))
{
    // You can decide if the origin in $_SERVER['HTTP_ORIGIN'] is something you want to allow, or as we do here, just allow all
    header("Access-Control-Allow-Origin: {$_SERVER['HTTP_ORIGIN']}");
}
else
{
    //No HTTP_ORIGIN set, so we allow any. You can disallow if needed here
    header("Access-Control-Allow-Origin: *");
}

header("Access-Control-Allow-Credentials: true");
header("Access-Control-Max-Age: 600");    // cache for 10 minutes

if($_SERVER["REQUEST_METHOD"] == "OPTIONS")
{
    if (isset($_SERVER["HTTP_ACCESS_CONTROL_REQUEST_METHOD"]))
        header("Access-Control-Allow-Methods: POST, GET, OPTIONS, DELETE, PUT"); //Make sure you remove those you do not want to support

    if (isset($_SERVER["HTTP_ACCESS_CONTROL_REQUEST_HEADERS"]))
        header("Access-Control-Allow-Headers: {$_SERVER['HTTP_ACCESS_CONTROL_REQUEST_HEADERS']}");

    //Just exit with 200 OK with the above headers for OPTIONS method
    exit(0);
}
//From here, handle the request as it is ok
```

Share Improve this answer Follow

answered Apr 19, 2017 at 14:15

Finn Johansen
331 ● 3 ● 3

This solves my issue - apparently my PHP webservice not able to entertain OPTIONS request properly - on which my Angular front end is relying upon prior to sending the POST request. Thanks! – Yazid Apr 27, 2021 at 14:09

29

I've simply managed to get dropzone and other plugin to work with this fix (angularjs + php backend)

```
 header('Access-Control-Allow-Origin: *');
 header("Access-Control-Allow-Credentials: true");
 header('Access-Control-Allow-Methods: GET, PUT, POST, DELETE, OPTIONS');
 header('Access-Control-Max-Age: 1000');
 header('Access-Control-Allow-Headers: Origin, Content-Type, X-Auth-Token , Authorization');
```

add this in your upload.php or where you would send your request (for example if you have upload.html and you need to attach the files to upload.php, then copy and paste these 4 lines). Also if you're using CORS plugins/addons in chrome/mozilla be sure to toggle them more than one time,in order for CORS to be enabled

Share

Improve this answer

Follow

edited May 9 at 3:37

Player1

3,135 ● 3 ● 29 ● 39

answered Nov 16, 2016 at 12:57

Fedeco

876 ● 10 ● 30

16

CORS can become a headache, if we do not correctly understand its functioning. I use them in PHP and they work without problems. reference here

```
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Credentials: true");
header("Access-Control-Max-Age: 1000");
header("Access-Control-Allow-Headers: X-Requested-With, Content-Type, Origin,
Cache-Control, Pragma, Authorization, Accept, Accept-Encoding");
header("Access-Control-Allow-Methods: PUT, POST, GET, OPTIONS, DELETE");
```

Share Improve this answer Follow

answered Jun 13, 2017 at 19:24

shades3002

981 ● 10 ● 12

1 I have used this in Codeigniter 4.1.3 and it doest work – Yougeshwar Khatri Sep 7, 2021 at 10:45

15

This much code works down for me when using angular 4 as the client side and PHP as the server side.

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

Share

Improve this answer

Follow

edited Dec 16, 2019 at 13:32

Machavity ♦

31.6k ● 27 ● 95 ● 105

answered Dec 14, 2017 at 9:45

Labib Hussain

590 ● 5 ● 9

1 Be careful while using '*' wildcard. Never open it unless that's what you really intend to do. As for testing your angular app specify localhost:4200 and it will work while still being safer. – Rohit Nair Mar 14, 2021 at 13:47

Tested on LAMP Server running PHP 7.4.x – Hugo Barbosa Nov 13, 2021 at 17:43

1 I was using Apache server and PHP backend language; I also fixed my problem with the same single line of code. – Sunil Sapkota Apr 10, 2023 at 9:06

10

this should work

```
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Headers: X-Requested-With, Content-Type, Origin,
Cache-Control, Pragma, Authorization, Accept, Accept-Encoding");
```

Share Improve this answer Follow

answered Jun 19, 2018 at 0:31

user8453321

344 ● 1 ● 5 ● 13

This worked really well on VUE + XAMPP (PHP) – Lepy Mar 27, 2022 at 20:57

6

I used these 5 headers and after that solved the cors error(backend: PHP, Frontend: VUE JS)

```
header('Access-Control-Allow-Origin: *');
header('Access-Control-Allow-Methods: GET, PUT, POST, DELETE, OPTIONS, post,
get');
header('Access-Control-Max-Age: 3600');
header('Access-Control-Allow-Headers: Origin, Content-Type, X-Auth-Token');
header('Access-Control-Allow-Credentials: true');
```

Share

Improve this answer

edited Oct 17, 2023 at 13:58

ruleboy21

6,333 ● 4 ● 21 ● 36

answered Apr 30, 2022 at 4:35

Mohamad Shabihi

130 ● 1 ● 5

Follow

3

add this code in **.htaccess**

add custom authentication key's in header like app_key,auth_key..etc

```
Header set Access-Control-Allow-Origin "*"
Header set Access-Control-Allow-Headers: "customKey1,customKey2, headers,
Origin, X-Requested-With, Content-Type, Accept, Authorization"
```

Share Improve this answer Follow

answered Jun 11, 2019 at 16:33

Rakyesh Kadadas

941 ● 9 ● 11

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.