

به نام کیمیاگر عالم



دانشگاه صنعتی امیر کبیر
(پلی تکنیک تهران)

امنیت شبکه

عنوان

فعالیت آزمایشگاهی – CORS

مدرس

دکتر سیاوش خرسندی

دانشجو

امیر حسین بابائیان

۴۰۱۱۳۱۰۰۲

ترم پاییز ۰۲-۰۱

گروه معماری کامپیوتر و شبکه های کامپیوتری

دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیر کبیر (پلی تکنیک تهران)

فهرست

فهرست	۲
تشریح CORS	۴
تعریف	۴
عملکرد	۴
کارکرد	۴
فواید	۴
معایب	۴
چرا استفاده می کنیم ؟	۵
توضیح CORS Response	۵
توضیحات CORS Request	۶
توضیحات Simple requests	۶
توضیحات Preflighted requests	۶
اضافه کردن CORS به برنامه Nodejs Express	۷
مراحل فعالیت	۷
دانلود فایل ها از git	۷
نصب npm و nodeJs	۷
افزودن نیازمندی ها	۸
اجرای dev	۸
سرور راه اندازی شده	۸

۹	فراخوانی API
۱۰	اجرای API با curl
۱۰	اضافه کردن CORS
۱۱	اجرای curl
۱۱	فراخوانی Call API در مرورگر
۱۱	اجبار در درخواست preflight
۱۲	اجرای Call API با مرورگر
۱۲	جزئیات درخواست اول
۱۳	جزئیات دستور دوم
۱۳	استفاده از CORS middleware برای Express
۱۳	نصب cors
۱۴	Network در مرورگر
۱۴	فعال سازی middleware
۱۵	خروجی نهایی

تشریح CORS

CORS (Cross-Origin Resource Sharing) یک مکانیزم مهم در امنیت وب است که به شما اجازه می‌دهد تا تصمیم بگیرید که درخواست‌هایی که از داخل دامنه‌های مختلف ارسال می‌شوند، پذیرفته شوند یا نه. در واقع، CORS یک روش برای بررسی درخواست‌های ارسال شده به سرور از داخل دامنه‌های مختلف است. اگر درخواست از داخل دامنه دیگری باشد، سرور با استفاده از پیام HTTP پاسخ به مرورگر خود می‌گوید که باید یا نه بتواند اطلاعات را ارسال کند. همچنین CORS برای حفاظت از اطلاعات شخصی کاربران بسیار مهم است. بدون این مکانیزم، سایت‌های دیگر می‌توانند بدون تأیید شما به اطلاعات شخصی شما دسترسی پیدا کنند.

تعریف

CORS به عنوان مکانیزم خودکار است که به سرورها اجازه می‌دهد تا درخواست‌های بین دامنه را پذیرش یا رد کنند.

عملکرد

با استفاده از CORS، سرور می‌تواند با توجه به درخواست خاص دامنه‌ای، درخواست‌های ارسالی از آن دامنه را پذیرش یا رد کند.

کارکرد

CORS با استفاده از پیام‌های HTTP، به سرور اعلام می‌کند که درخواست دارای مجوز است یا خیر و سرور می‌تواند بر اساس این عمل کند.

فواید

با استفاده از CORS، سرور می‌تواند تنها درخواست‌های مجاز از دامنه‌های مختلف را پذیرش کرده و با ایجاد امنیت بیشتر، سایت خود را در برابر دسترسی‌های غیر مجاز محافظت نماید.

معایب

ممکن است بعضی مرورگرها از CORS به صورت کامل پشتیبانی نکنند و همچنین بعضا باعث کاهش کارایی شود.

چرا استفاده می کنیم ؟

بیشتر زمان ها، اسکریپتی که در مرورگر کاربر اجرا می شود فقط به منابع با محل اصلی مشابه (به عنوان مثال API فراخوانی به همان بک اند به همان نحوه کد جاوا اسکریپت داده شده) نیاز دارد. بنابراین، این عدم قابلیت دسترسی عادی جاوا اسکریپت به منابع در محل اصلی دیگر به عنوان چیزی خوب برای امنیت شناخته می شود.

در این مفاهیم، "محل اصلی دیگر" به معنی آدرس URL در حال دسترسی از محلی که جاوا اسکریپت اجرا می شود، با داشتن:

- یک طرح مختلف (HTTP یا HTTPS)
- یک دامنه مختلف
- یک پورت مختلف

اگر شما در حال اجرای SPA ری اکت که فراخوانی به یک بک اند API بر روی یک دامنه مختلف اجرا می شود، فونت های وب نیز بر روی CORS به کار می روند.

توضیح CORS Response

توضیحات در مورد پاسخ CORS (Cross-Origin Resource Sharing) می توان به سه روش مشخص کرد. نخستین روش شناسایی پاسخ CORS با بررسی **سرآیندهای پاسخ HTTP** است. سرور سرآیندهای خاصی مانند سرآیند Access-Control-Allow-Origin را به عنوان نشان دهنده اجازه دادن برای درخواست ها از منبع مشخصی اضافه می کند. مرورگر همچنین می تواند برای سرآیندهای مشخص، مانند سرآیند Access-Control-Allow-Methods، برای تعیین روش هایی که سرور برای درخواست ها به منابع مختلف مجاز است، بگردد.

روش دیگر شناسایی پاسخ CORS با نوع **کد وضعیت HTTP** است که سرور بازگردانده. به عنوان مثال، اگر سرور یک کد وضعیت مانند ۲۰۰ OK بازگرداند، آن به معنی موفقیت در درخواست است. در حالت دیگر، اگر سرور یک کد وضعیت مانند ۴۰۴ Not Found بازگرداند، آن به معنی یافت نشدن منبع است. در صورت بوجود خطای مربوط به CORS، سرور ممکن است یک کد وضعیت مانند ۵۰۰ Internal Server Error بدهد.

به همراه این روش ها، ابزار توسعه دهنده مرورگر هم می تواند اطلاعات درباره پاسخ CORS ، شامل هدرها و کدهای وضعیت، را فراهم آورد. این می تواند برای اشکال زدایی و درک جزئیات پاسخ CORS مفید باشد.

توضیحات CORS Request

CORS به دو دسته عملیات مختلف در پاسخ به درخواست ها برای عبور منابع مختلف مراجعه می کند، که به عنوان درخواست های معمولی (simple requests) و درخواست های پیچیده (preflighted requests) شناخته می شوند.

درخواست های معمولی شامل ارسال داده ها از نوع آنها با استفاده از HTTP GET و HTTP POST هستند. در این صورت، مرورگر به عنوان خود از درخواست بدون پیش فرض اطمینان می گیرد و پاسخ به درخواست را ارسال می کند.

درخواست های پیچیده شامل درخواست های HTTP مختلف با استفاده از HTTP متفاوت از GET و POST ، به عنوان مثال HTTP PUT ، HTTP DELETE و HTTP PATCH هستند. در این صورت، مرورگر قبل از ارسال داده ها، برای تأیید پذیرش درخواست با پاسخ از سوی سرور، یک درخواست preflight به سرور ارسال می کند.

توضیحات Simple requests

در زبان CORS ، درخواست های ساده یا GET ، POST و HEAD به معنای درخواست های خودکارانداز شده به سرور برای گرفتن داده های مورد نظر است. این درخواست ها در حالت عادی از بدوی درخواست کننده به سرور فرستاده می شود و تعدادی از هدرهای مربوط به توضیحات به صورت خودکار اضافه می شود؛ اما در حالت های خاص، ممکن است برای فرستادن داده های بیشتر به سرور، هدرهای خاصی باید تعیین کنید. درخواست های GET از سرور داده های مورد نظر را بدون تغییر بدست می آورند، درخواست های POST برای فرستادن داده ها به سرور به کار می روند و درخواست های HEAD داده های مربوط به جواب درخواست را بدون اطلاعات داده ای بدست می آورند.

توضیحات Preflighted requests

در CORS ، درخواست های Preflight به عنوان درخواست های OPTIONS نامیده می شوند. این درخواست ها قبل از اجرای درخواست خود، به عنوان درخواست آزمایشی به سرور ارسال می شوند و برای بررسی امکانات

سرور برای اجرای درخواست خود استفاده می شود. در این درخواست، باید سرور پاسخ خاصی با هدر های مشخصی برای نشان دادن مجوز ها و قوانین مربوط به درخواست برای اجرا به مرورگر برگرداند. این پروسه های آزمایشی به منظور جلوگیری از اجرای درخواست های غیرمجاز است که می تواند به سرور آسیب برساند.

اضافه کردن CORS به برنامه Nodejs Express

به عنوان مثال طبیعی از کارکرد این، برنامه Node Express موجود را تغییر دهید و به درخواست های JavaScript بین دو دامنه برای آن فعال کنید. برنامه که با استفاده از ۳ Vue CLI ایجاد شده و در پورت ۳۰۰۰ اجرا می شود، با یک سرور Express در پورت ۳۰۰۱ اجرا می شود. رابط کاربری Vue فراهم می کند که یک تماس API به سرور داشته باشد، اما متأسفانه این کار، کار نمی کند زیرا سرور CORS فعال نیست، در این فایل قرار است این مورد را حل کنیم.

مراحل فعالیت

در این بخش همراه با تصویر از عملیات ها اقدامات انجام شده روی kali Linux را نمایش می دهیم.

دانلود فایل ها از git

```
(kali@kali)-[~/Desktop/AHB/NetworkSecurity]
$ git clone https://github.com/auth0-blog/express-cors-demo.git
Cloning into 'express-cors-demo' ...
remote: Enumerating objects: 63, done.
remote: Total 63 (delta 0), reused 0 (delta 0), pack-reused 63
Receiving objects: 100% (63/63), 127.02 KiB | 172.00 KiB/s, done.
Resolving deltas: 100% (21/21), done.
```

نصب npm و nodejs

```
(kali@kali)-[~/Desktop/AHB/NetworkSecurity]
$ sudo apt-get install npm
[sudo] password for kali:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
dh-elpa-helper docutils-common python3-alabaster python3-docutils
```

افزودن نیازمندی ها

متأسفانه به ارور ناشناخته ای برخورد کردیم و با جستجو نیز مشکل حل نشد در کالی، احتمالاً مشکل به وجود آمده به دلیل ماشین مجازی و شبکه ی تعریف شده اش بود، پس از این بخش به بعد در ویندوز فعالیت ادامه می یابد، اسکرین شات های تکراری قرار داده نمی شوند.

اجرای dev

```
Select npm

DONE Compiled successfully in 16608ms

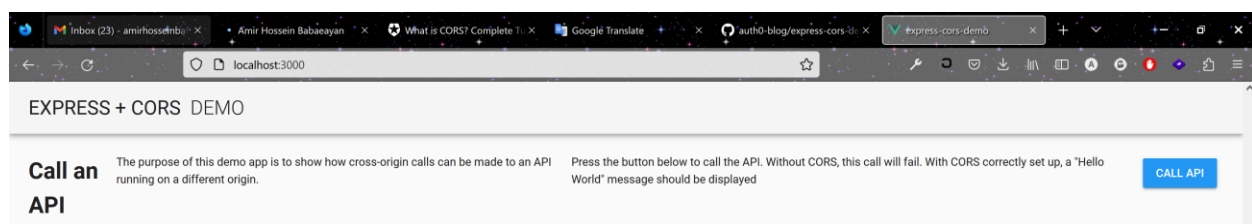
App running at:
- Local: http://localhost:3000/
- Network: http://192.168.56.1:3000/

Note that the development build is not optimized.
To create a production build, run npm run build.
```

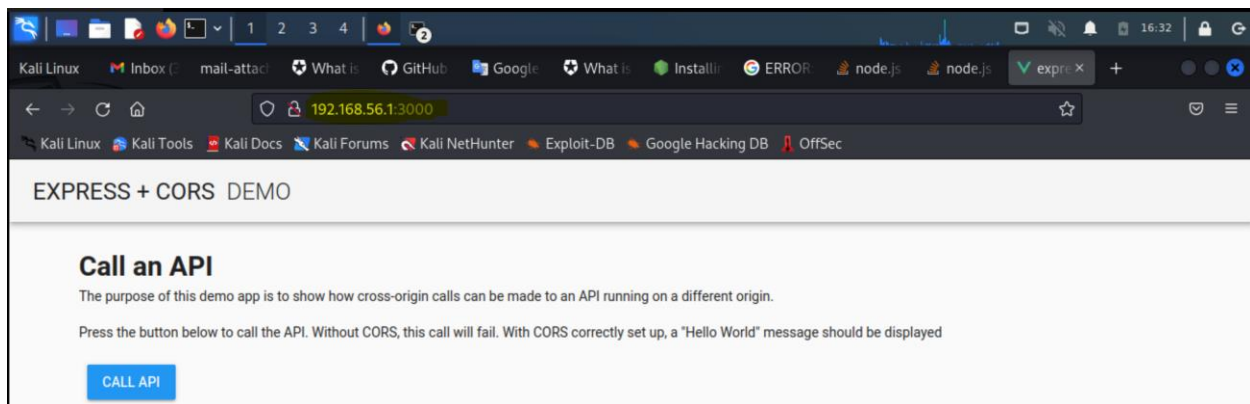
با موفقیت در ویندوز انجام شد و مشکل قبلی در مجوز های فایروال بود که در این جا پس از درخواست به صورت گرافیکی داده شد.

سرور راه اندازی شده

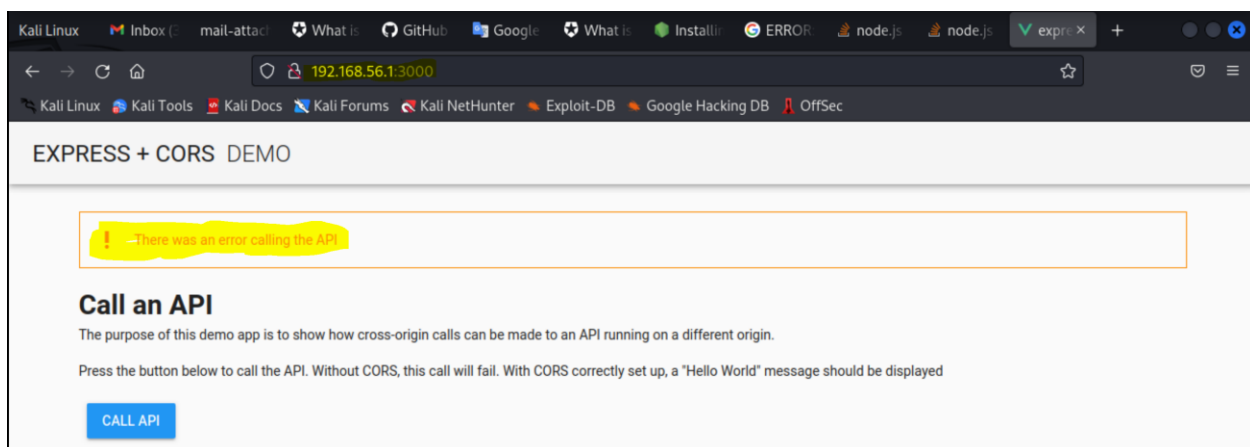
به آدرس localhost:3000 در مرورگر می رویم.



در کالی نیز با استفاده از آدرس شبکه یعنی ۱۹۲,۱۶۸,۵۶,۱:۳۰۰۰ که داده شد دسترسی یافتیم.



فراخوانی API

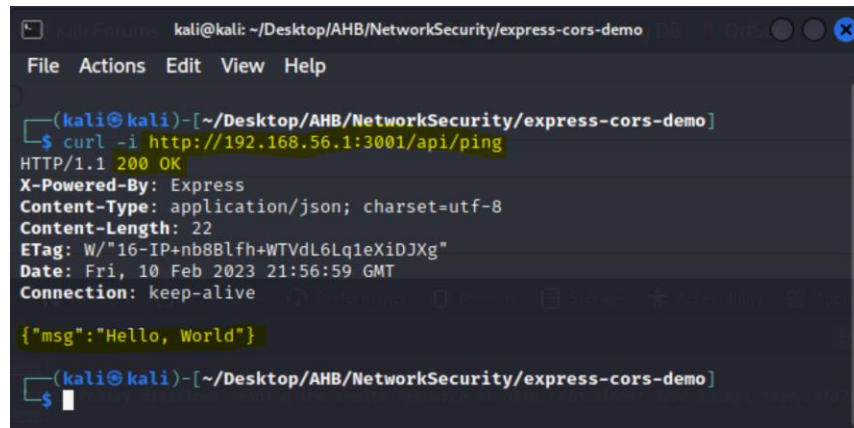


همانطور که در توضیحات اشاره شده بایستی با خطا مواجه شود که مواجه شدیم، شرح بیشتر مشکل در تصویر آورده شده است:

❌ Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at <http://localhost:3001/api/ping>. (Reason: CORS request did not succeed). Status code: (null). [\[Learn More\]](#)

این خطا به این دلیل است که به API در آدرس `http://localhost:3001/api/ping` یک درخواست داده می شود، که با عدد پورت متفاوتی می باشد که پورت اصلی برنامه وب آن تفاوت دارد. بنابراین به صورت پیش فرض توسط مرورگر مسدود می شود.

اجرای API با curl



```
kali@kali: ~/Desktop/AHB/NetworkSecurity/express-cors-demo
File Actions Edit View Help

(kali@kali)-[~/Desktop/AHB/NetworkSecurity/express-cors-demo]
$ curl -i http://192.168.56.1:3001/api/ping
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 22
ETag: W/"16-IP+nb8Blfh+WTvDL6Lq1eXiDJXg"
Date: Fri, 10 Feb 2023 21:56:59 GMT
Connection: keep-alive

{"msg":"Hello, World"}

(kali@kali)-[~/Desktop/AHB/NetworkSecurity/express-cors-demo]
$
```

زمانی که از curl استفاده کردیم، دیگر به خطا برخوردیم چرا که CORS فقط بر روی XMLHttpRequest مرورگر فراخوانی می شود.

اضافه کردن CORS

کد index.js داخل پوشه server را تغییر می دهیم که فایل تغییر داده شده به شرح ذیل است:

```
// Refrence: https://auth0.com/blog/cors-tutorial-a-guide-to-cross-origin-resource-sharing/
const express = require("express");
const debug = require("debug")("server");

const app = express();
const port = process.env.SERVER_PORT || 3001;

// NEW - Add CORS headers - see https://enable-cors.org/server\_expressjs.html
app.use(function(req, res, next) {
  res.header("Access-Control-Allow-Origin", "*");
  res.header(
    "Access-Control-Allow-Headers",
    "Origin, X-Requested-With, Content-Type, Accept"
  );
  next();
});

// API endpoint
app.get("/api/ping", (req, res) => {
  res.send({
    msg: "Hello, World"
  });
});
```

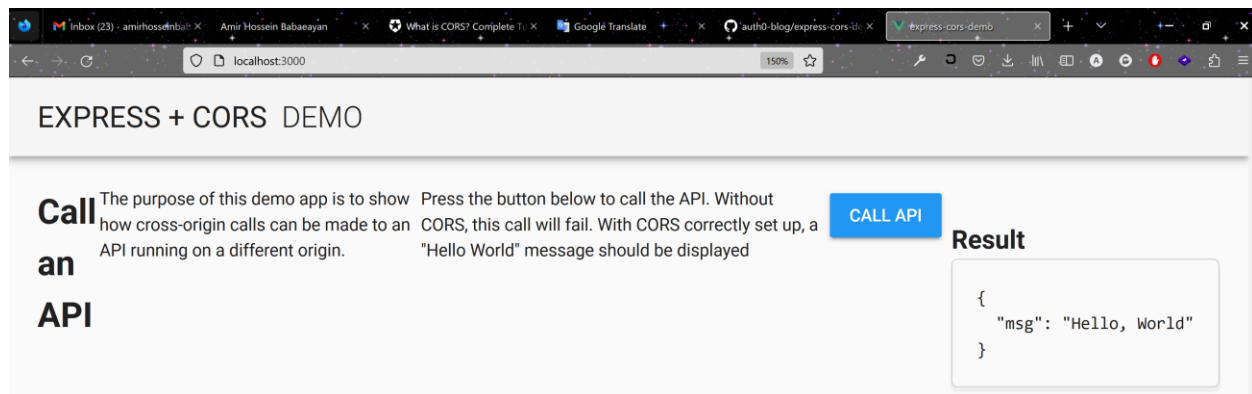
```
});  
  
app.listen(port, () => debug(`Listening on port ${port}`));
```

اجرای curl

```
kali@kali: ~/Desktop/AHB/NetworkSecurity/express-cors-demo
File Actions Edit View Help
$ curl -i http://192.168.56.1:3001/api/ping
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Content-Type: application/json; charset=utf-8
Content-Length: 22
ETag: W/"16-IP+nb8Blfh+WTvDL6Lq1eXiDJXg"
Date: Fri, 10 Feb 2023 22:10:44 GMT
Connection: keep-alive

{"msg": "Hello, World"}
```

فرخوانی Call API در مرورگر



مشاهده می شود که مشکل حل شده است و نتیجه نمایش داده می شود.

اجبار در درخواست preflight

قطعه کد موجود در لیک را به فایل CallApi.vue اضافه میکنیم تا یک درخواست غیر ساده باشد، شرح تغییرات را می توان در کد مشاهده کرد. (با رنگ قرمز مشخص شده است.)

```
methods: {  
  callApi() {  
    fetch("//localhost:3001/api/ping", {headers:{"Content-Type": "application/json"}})  
    .then(async response => {  
      if (response.ok) {  
        this.apiError = false;  
        this.result = await response.json();  
      }  
    })  
  }  
}
```

```

    } else {
      this.apiError = true;
    }
  })
  .catch(() => (this.apiError = true));
}
}

```

اجرای Call API با مرورگر

EXPRESS + CORS DEMO

The purpose of this demo app is to show how cross-origin calls can be made to an API running on a different origin. Press the button below to call the API. Without CORS, this call will fail. With CORS correctly set up, a "Hello World" message should be displayed

CALL API

Result

```

{
  "msg": "Hello, World"
}

```

Status	Method	Domain	File	Initiator	Type	Transferred	Size	0 ms	80 ms	160 ms
200	OPTIONS	localhost:3001	ping	fetch	html	338 B	8 B	4 ms		
304	GET	localhost:3001	ping	qs.js:48 (fetch)	json	cached	22 B	1 ms		

میبینید که ابتدا یک دستور preflight داده است و سپس اقدامات مربوط به ping انجام شده است.

جزئیات درخواست اول

Headers

Copy

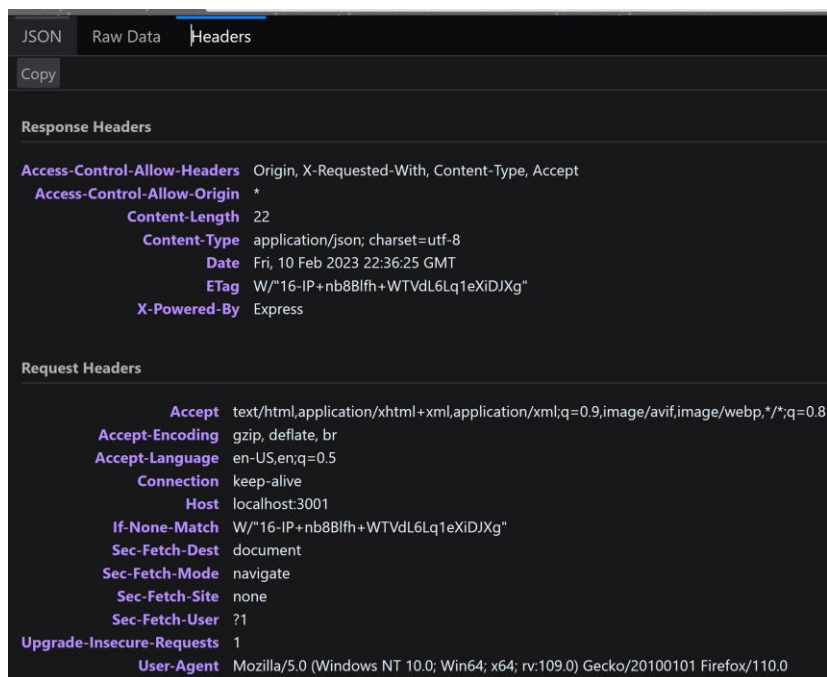
Response Headers

- Access-Control-Allow-Headers** Origin, X-Requested-With, Content-Type, Accept
- Access-Control-Allow-Origin** *
- Content-Length** 22
- Content-Type** application/json; charset=utf-8
- Date** Fri, 10 Feb 2023 22:34:48 GMT
- ETag** W/"16-IP+nb8Blfh+WTVDL6Lq1eXiDJXg"
- X-Powered-By** Express

Request Headers

- Accept** text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
- Accept-Encoding** gzip, deflate, br
- Accept-Language** en-US,en;q=0.5
- Connection** keep-alive
- Host** localhost:3001
- If-None-Match** W/"16-IP+nb8Blfh+WTVDL6Lq1eXiDJXg"
- Sec-Fetch-Dest** document
- Sec-Fetch-Mode** navigate
- Sec-Fetch-Site** cross-site
- Upgrade-Insecure-Requests** 1
- User-Agent** Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/110.0

جزئیات دستور دوم



استفاده از CORS middleware برای Express

از پکیج های موجود بجای تنظیم دستی استفاده می شود.

نصب cors

```
PS D:\Learning\4011\NetworkSecurity\p_webSec\install\express-cors-demo> npm install cors
npm WARN SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\watchpack\node_modules\fsevents):
npm WARN SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm WARN SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.7 (node_modules\fsevents):
npm WARN SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.7: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
+ cors@2.8.5
added 1 package from 1 contributor, updated 1 package and audited 1405 packages in 25.618s
found 76 vulnerabilities (9 low, 22 moderate, 33 high, 12 critical)
run npm audit fix to fix them, or npm audit for details
```

اضافه کردن CORS

حال کد index.js را به شکل زیر تغییر می دهیم.

```
const express = require("express");
const debug = require("debug")("server");

// NEW - bring in the cors library
const cors = require("cors");

const app = express();
const port = process.env.SERVER_PORT || 3001;

// NEW - replace custom middleware with the cors() middleware
```

```
app.use(cors());

app.get("/api/ping", (req, res) => {
  res.send({
    msg: "Hello, World"
  });
});

app.listen(port, () => debug(`Listening on port ${port}`));
```

Network در مرورگر

The screenshot shows a web browser at localhost:3000 displaying an "EXPRESS + CORS DEMO". The page has a text area with instructions, a "CALL API" button, and a "Result" box showing the response: `{ "msg": "Hello, World" }`.

Below the browser window, the Chrome DevTools Network tab is open, showing a list of requests. The selected request is an "OPTIONS" request to localhost:3001/ping. The "Headers" sub-tab is active, showing the "Access-Control-Allow-Headers" header with the value "content-type". Other headers include "Access-Control-Allow-Methods", "Access-Control-Allow-Origin", "Connection", "Content-Length", "Date", and "Vary".

فعال سازی middleware

```
app.use(
  cors({
    origin: "http://localhost:3000", // restrict calls to those this address
    methods: "GET" // only allow GET requests
  })
);
```

قطعه کد فوق را جایگزین قطعه کد ذیل در index.js می کنیم.

```
app.use(cors());
```

حال خروجی نهایی را در مرورگر مشاهده می کنیم.

EXPRESS + CORS DEMO

The purpose of this demo app is to show how cross-origin calls can be made to an API running on a different origin. Press the button below to call the API. Without CORS, this call will fail. With CORS correctly set up, a "Hello World" message should be displayed

CALL API

Result

```
{
  "msg": "Hello, World"
}
```

204 OPTIONS localhost:3001 ping fetch plain 307 B 0 B

304 GET localhost:3001 ping json cached 22 B

2 requests 22 B / 307 B transferred Finish: 21 ms

Access-Control-Allow-Headers: content-type
Access-Control-Allow-Methods: GET
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 0
Date: Fri, 10 Feb 2023 12:58:11 GMT
Vary: Origin, Access-Control-Request-Headers

amirhosseinbabayan 48113100

مشاهده می شود که سیاست اعمال شده یعنی صرفا GET مجاز باشد فعال شده است آن هم با استفاده از cors تعریف شده به صورت سیستمی.