

Detalhes do pacote

cors

EX **EXPRESSJS** 🔥 10.8M MIT 2.8.5

Node.js CORS middleware

⇒ [cors](#), [express](#), [connect](#), [middleware](#)

readme (leia-me)

cors

npm v2.8.5 downloads 11M/month build passing coverage 100%

CORS is a node.js package for providing a [Connect/Express](#) middleware that can be used to enable **CORS** with various options.

Follow me (@troygoode) on Twitter!

- [Installation](#)
- [Usage](#)
 - [Simple Usage](#)
 - [Enable CORS for a Single Route](#)
 - [Configuring CORS](#)
 - [Configuring CORS Asynchronously](#)
 - [Enabling CORS Pre-Flight](#)
- [Configuration Options](#)
- [Demo](#)
- [License](#)
- [Author](#)

Installation

This is a [Node.js](#) module available through the [npm](#) registry. Installation is done using the [npm install](#) command:

```
$ npm install cors
```

Usage

Simple Usage (Enable *All* CORS Requests)

```
var express = require('express')
var cors = require('cors')
var app = express()

app.use(cors())

app.get('/products/:id', function (req, res, next) {
  res.json({msg: 'This is CORS-enabled for all origins!'})
})

app.listen(80, function () {
  console.log('CORS-enabled web server listening on port 80')
})
```

Enable CORS for a Single Route

```
var express = require('express')
var cors = require('cors')
var app = express()

app.get('/products/:id', cors(), function (req, res, next) {
  res.json({msg: 'This is CORS-enabled for a Single Route'})
})

app.listen(80, function () {
  console.log('CORS-enabled web server listening on port 80')
})
```

Configuring CORS

```
var express = require('express')
var cors = require('cors')
var app = express()

var corsOptions = {
  origin: 'http://example.com',
  optionsSuccessStatus: 200 // some legacy browsers (IE11, various SmartTVs) choke on 204
}

app.get('/products/:id', cors(corsOptions), function (req, res, next) {
  res.json({msg: 'This is CORS-enabled for only example.com.'})
})

app.listen(80, function () {
  console.log('CORS-enabled web server listening on port 80')
})
```

Configuring CORS w/ Dynamic Origin

```
var express = require('express')
var cors = require('cors')
var app = express()

var whitelist = ['http://example1.com', 'http://example2.com']
var corsOptions = {
  origin: function (origin, callback) {
    if (whitelist.indexOf(origin) !== -1) {
      callback(null, true)
    } else {
      callback(new Error('Not allowed by CORS'))
    }
  }
}

app.get('/products/:id', cors(corsOptions), function (req, res, next) {
  res.json({msg: 'This is CORS-enabled for a whitelisted domain.'})
})

app.listen(80, function () {
  console.log('CORS-enabled web server listening on port 80')
})
```

If you do not want to block REST tools or server-to-server requests, add a `!origin` check in the origin function like so:

```
var corsOptions = {
  origin: function (origin, callback) {
    if (whitelist.indexOf(origin) !== -1 || !origin) {
      callback(null, true)
    } else {
      callback(new Error('Not allowed by CORS'))
    }
  }
}
```

Enabling CORS Pre-Flight

Certain CORS requests are considered 'complex' and require an initial **OPTIONS** request (called the "pre-flight request"). An example of a 'complex' CORS request is one that uses an HTTP verb other than GET/HEAD/POST (such as DELETE) or that uses custom headers. To enable pre-flighting, you must add a new OPTIONS handler for the route you want to support:

```
var express = require('express')
var cors = require('cors')
var app = express()

app.options('/products/:id', cors()) // enable pre-flight request for DELETE request
app.del('/products/:id', cors(), function (req, res, next) {
  res.json({msg: 'This is CORS-enabled for all origins!'})
})

app.listen(80, function () {
  console.log('CORS-enabled web server listening on port 80')
})
```

You can also enable pre-flight across-the-board like so:

```
app.options('*', cors()) // include before other routes
```

Configuring CORS Asynchronously

```
var express = require('express')
var cors = require('cors')
var app = express()

var whitelist = ['http://example1.com', 'http://example2.com']
var corsOptionsDelegate = function (req, callback) {
  var corsOptions;
  if (whitelist.indexOf(req.header('Origin')) !== -1) {
    corsOptions = { origin: true } // reflect (enable) the requested origin in the CO
  } else {
    corsOptions = { origin: false } // disable CORS for this request
  }
  callback(null, corsOptions) // callback expects two parameters: error and options
}

app.get('/products/:id', cors(corsOptionsDelegate), function (req, res, next) {
  res.json({msg: 'This is CORS-enabled for a whitelisted domain.'})
})

app.listen(80, function () {
  console.log('CORS-enabled web server listening on port 80')
})
```

Configuration Options

- **origin** : Configures the **Access-Control-Allow-Origin** CORS header. Possible values:
 - **Boolean** - set **origin** to **true** to reflect the **request origin**, as defined by `req.header('Origin')` , or set it to **false** to disable CORS.
 - **String** -set **origin** to a specific origin. For example if you set it to `"http://example.com"` only requests from `"http://example.com"` will be allowed.
 - **RegExp** - set **origin** to a regular expression pattern which will be used to test the request origin. If it's a match, the request origin will be reflected. For example the pattern `/example\.com$/` will reflect any request that is coming from an origin ending with `"example.com"`.
 - **Array** - set **origin** to an array of valid origins. Each origin can be a **String** or a **RegExp**. For example `["http://example1.com", /\.example2\.com$/]` will accept any request from `"http://example1.com"` or from a subdomain of `"example2.com"`.
 - **Function** - set **origin** to a function implementing some custom logic. The function takes the request origin as the first parameter and a callback (which expects the signature `err [object], allow [bool]`) as the second.
- **methods** : Configures the **Access-Control-Allow-Methods** CORS header. Expects a comma-delimited string (ex: 'GET,PUT,POST') or an array (ex: `['GET', 'PUT', 'POST']`).
- **allowedHeaders** : Configures the **Access-Control-Allow-Headers** CORS header. Expects a comma-delimited string (ex: 'Content-Type,Authorization') or an array (ex: `['Content-Type', 'Authorization']`). If not specified, defaults to reflecting the headers specified in the request's **Access-Control-Request-Headers** header.
- **exposedHeaders** : Configures the **Access-Control-Expose-Headers** CORS header. Expects a comma-delimited string (ex: 'Content-Range,X-Content-Range') or an array (ex: `['Content-Range', 'X-Content-Range']`). If not specified, no custom headers are exposed.
- **credentials** : Configures the **Access-Control-Allow-Credentials** CORS header. Set to **true** to pass the header, otherwise it is omitted.
- **maxAge** : Configures the **Access-Control-Max-Age** CORS header. Set to an integer to pass the header, otherwise it is omitted.
- **preflightContinue** : Pass the CORS preflight response to the next handler.
- **optionsSuccessStatus** : Provides a status code to use for successful **OPTIONS** requests, since some legacy browsers (IE11, various SmartTVs) choke on `204` .

The default configuration is the equivalent of:

```
{
  "origin": "*",
  "methods": "GET,HEAD,PUT,PATCH,POST,DELETE",
  "preflightContinue": false,
  "optionsSuccessStatus": 204
}
```

For details on the effect of each CORS header, read [this](#) article on HTML5 Rocks.

Demo

A demo that illustrates CORS working (and not working) using jQuery is available here: <http://node-cors-client.herokuapp.com/>

Code for that demo can be found here:

- Client: <https://github.com/TroyGoode/node-cors-client>
- Server: <https://github.com/TroyGoode/node-cors-server>


License

[MIT License](#)

Author

Troy Goode (troygoode@gmail.com)

Recolher ▾

yarn.pm/cors 

expressjs/cors

npm cors

Use

\$ yarn add cors 

Experimente no RunKit · [Navegar pelos arquivos](#)

CDNs

unpkg [unpkg.com/cors/](#)
jsDelivr [cdn.jsdelivr.net/npm/cors/](#)
bundle.run [bundle.run/cors](#)

Popularidade

☆ Estrelas no GitHub **4.262**
☁ Downloads dos últimos 30 dias **10.8m**
☁ jsDelivr last 30 days **669**
🔗 Dependentes **0**

Atividade

Commits dos últimos 3 meses **0**
🕒 Último commit **27 semanas atrás**

Utilização

📦 Dependências **2**
📦 DevDependencies **6**
📦 Pacotes [ver package.json](#)
📦 Tamanho no navegador **4.31KB**

Tags

latest **2.8.5**

Versões

4 de novembro de 2018 **2.8.5**
12 de julho de 2017 **2.8.4**
29 de março de 2017 **2.8.3**

Mostrar tudo ▾

Contribuidores

 DOUGWILSON
 TROYGOODE