Docs » Flask-CORS

Flask-CORS

build passing | pypi v3.0.10 | python 2.7 | 3.4 | 3.5 | 3.6 | 3.7 | license mit

A Flask extension for handling Cross Origin Resource Sharing (CORS), making cross-origin AJAX possible.

This package has a simple philosophy: when you want to enable CORS, you wish to enable it for all use cases on a domain. This means no mucking around with different allowed headers, methods, etc.

By default, submission of cookies across domains is disabled due to the security implications. Please see the documentation for how to enable credential'ed requests, and please make sure you add some sort of CSRF protection before doing so!

Installation

Install the extension with using pip, or easy_install.

```
$ pip install -U flask-cors
```

Usage

This package exposes a Flask extension which by default enables CORS support on all routes, for all origins and methods. It allows parameterization of all CORS headers on a per-resource level. The package also contains a decorator, for those who prefer this approach.

Simple Usage

In the simplest case, initialize the Flask-Cors extension with default arguments in order to allow CORS for all domains on all routes. See the full list of options in the documentation.

```
from flask import Flask
from flask_cors import CORS

app = Flask(__name__)
CORS(app)

@app.route("/")
def helloWorld():
  return "Hello, cross-origin-world!"
```

Resource specific CORS

Alternatively, you can specify CORS options on a resource and origin level of granularity by passing a dictionary as the *resources* option, mapping paths to a set of options. See the full list of options in the documentation.

```
app = Flask(__name__)
cors = CORS(app, resources={r"/api/*": {"origins": "*"}})

@app.route("/api/v1/users")
def list_users():
  return "user example"
```

Route specific CORS via decorator

This extension also exposes a simple decorator to decorate flask routes with. Simply add `@cross_origin()` below a call to Flask's `@app.route(..)` to allow CORS on a given route. See the full list of options in the decorator documentation.

```
@app.route("/")
@cross_origin()
def helloWorld():
  return "Hello, cross-origin-world!"
```

Documentation

For a full list of options, please see the full documentation

Troubleshooting

If things aren't working as you expect, enable logging to help understand what is going on under the hood, and why.

```
logging.getLogger('flask_cors').level = logging.DEBUG
```

Tests

A simple set of tests is included in `test/`. To run, install nose, and simply invoke `nosetests` or `python setup.py test` to exercise the tests.

Contributing

Questions, comments or improvements? Please create an issue on Github, tweet at @corydolphin or send me an email. I do my best to include every contribution proposed in any way that I can.

Credits

This Flask extension is based upon the Decorator for the HTTP Access Control written by Armin Ronacher.