Flask-Cors Documentation

Release 1.5.0

Cory Dolphin

August 22, 2014

Flask-CORS

Flask-CORS is a simple extension to Flask allowing you to support cross origin resource sharing (CORS) using a simple decorator.

[![Build Status](https://travis-ci.org/wcdolphin/flask-cors.png?branch=master)](https://travis-ci.org/wcdolphin/flask-cors)

Installation

Install the extension with using pip, or easy_install. `bash $ pip install flask-cors `

Usage This extension exposes a simple decorator to decorate flask routes with. Simply add *@cross_origin()* below a call to Flask's *@app.route(..)* incanation to accept the default options and allow CORS on a given route.

Simple Usage

```python @app.route("/") @cross_origin() # allow all origins all methods. def helloWorld():
    return "Hello, cross-origin-world!"
```

Using JSON with Cross Origin When using JSON cross origin, browsers will issue a pre-flight OPTIONS request for POST requests. In order for browsers to allow POST requests with a JSON content type, you must allow the Content-Type header.

```python @app.route("/user/create", methods=['GET','POST']) @cross_origin(headers=['Content-Type']) # Send Access-Control-Allow-Headers def cross_origin_json_post():
    return jsonify(success=True)
```

Application-wide settings Alternatively, you can set any of these options in an app's config object. Setting these at the application level effectively changes the default value for your application, while still allowing you to override it on a per-resource basis.

The application-wide configuration options are creatively prefixed with **CORS_** e.g. * CORS_ORIGINS * CORS_METHODS * CORS_HEADERS * CORS_EXPOSE_HEADERS * CORS_ALWAYS_SEND * CORS_MAX_AGE * CORS_SEND_WILDCARD * CORS_ALWAYS_SEND * CORS_AUTOMATIC_OPTIONS

```python app.config['CORS_ORIGINS'] = ['https://foo.com', 'http://www.bar.com'] app.config['CORS_HEADERS'] = 'Content-Type'
```

Will return CORS headers for origins 'https://foo.com'and 'http://www.bar.com' # and an Access-Control-Allow-Headers of 'Content-Type' """ Will return CORS headers for origins 'https://foo.com'and 'http://www.bar.com' and an Access-Control-Allow-Headers of 'Content-Type' E.G. Testing with httpie

    ~ http GET http://127.0.0.1:5000/ HTTP/1.0 200 OK Access-Control-Allow-Headers: Content-Type Access-Control-Allow-Origin: https://foo.com, http://www.bar.com Content-Length: 26 Content-Type: text/html; charset=utf-8 Date: Tue, 22 Jul 2014 23:55:53 GMT Server: Werkzeug/0.9.4 Python/2.7.8

    Hello, cross-origin-world!

""" @app.route("/") @cross_origin() def helloWorld():

    return "Hello, cross-origin-world!"

""" Will return CORS headers for 'google.com' and Access-Control-Allow-Headers of 'Content-Type'. E.G. Testing with httpie

        ~ http GET http://127.0.0.1:5000/special

HTTP/1.0 200 OK Access-Control-Allow-Headers: Content-Type Access-Control-Allow-Origin: http://google.com Content-Length: 33 Content-Type: text/html; charset=utf-8 Date: Tue, 22 Jul 2014 23:55:29 GMT Server: Werkzeug/0.9.4 Python/2.7.8

Hello, google-cross-origin-world!

“”” @app.route(“/special”) @cross_origin(origins=”http://google.com”) def helloGoogle():

return “Hello, google-cross-origin-world!”

“‘

flask_cors.**cross_origin**(*origins=None*, *methods=None*, *headers=None*, *expose_headers=None*, *supports_credentials=False*, *max_age=None*, *send_wildcard=True*, *always_send=True*, *automatic_options=True*)

This function is the decorator which is used to wrap a Flask route with. In the simplest case, simply use the default parameters to allow all origins in what is the most permissive configuration. If this method modifies state or performs authentication which may be brute-forced, you should add some degree of protection, for example Cross Site Forgery Request protection.

Parameters

- **origins** (*list or string*) – The origin, or list of origins which are to be allowed, and injected into the returned *Access-Control-Allow-Origin* header
- **methods** (*list*) – The methods to be allowed and injected as the *Access-Control-Allow-Methods* header returned.
- **headers** (*list or string*) – The list of allowed headers to be injected as the *Access-Control-Allow-Headers* header returned.
- **expose_headers** – The list of headers to be exposed to browsers through the *Access-Control-Expose-Headers* header returned.
- **supports_credentials** (*bool*) – Allows users to make authenticated requests. If true, injects the *Access-Control-Allow-Credentials* header in responses.

  Note: this option cannot be used in conjuction with a ‘*’ origin
- **max_age** (*timedelta, integer, string or None*) – The maximum time for which this CORS request maybe cached. This value is set as the *Access-Control-Max-Age* header.
- **send_wildcard** (*bool*) – If True, and the origins parameter is *, a wildcard *Access-Control-Allow-Origin* header is sent, rather than echoing the request’s *Origin* header.
- **always_send** (*bool*) – If True, CORS headers are sent even if there is no *Origin* in the request’s headers.
- **automatic_options** (*bool*) – If True, CORS headers will be returned for OPTIONS requests. For use with cross domain POST requests which preflight OPTIONS requests, you will need to specifically allow the Content-Type header.