# GitHub-Flask Documentation

*Release 3.2.0*

**Cenk Altı**

**Jul 01, 2018**

# Contents

GitHub-Flask is an extension to Flask that allows you authenticate your users via GitHub using OAuth protocol and call GitHub API methods.

GitHub-Flask depends on the requests library.

CHAPTER 1

# Installation

Install the extension with the following command:

```
$ pip install GitHub-Flask
```

# CHAPTER 2

# Configuration

Here's an example of how GitHub-Flask is typically initialized and configured:

```
from flask import Flask
from flask_github import GitHub

app = Flask(__name__)
app.config['GITHUB_CLIENT_ID'] = 'XXX'
app.config['GITHUB_CLIENT_SECRET'] = 'YYY'

# For GitHub Enterprise
app.config['GITHUB_BASE_URL'] = 'https://HOSTNAME/api/v3/'
app.config['GITHUB_AUTH_URL'] = 'https://HOSTNAME/login/oauth/'

github = GitHub(app)
```

The following configuration settings exist for GitHub-Flask:

| | |
|---|---|
| *GITHUB_CLIENT_ID* | Your GitHub application's client id. Go to https://github.com/settings/applications to register new application. |
| *GITHUB_CLIENT_SECRET* | Your GitHub application's client secret. |
| *GITHUB_BASE_URL* | Base URL for API requests. Override this to use with GitHub Enterprise. Default is "https://api.github.com/". |
| *GITHUB_AUTH_URL* | Base authentication endpoint. Override this to use with GitHub Enterprise. Default is "https://github.com/login/oauth/". |

# CHAPTER 3

## Authenticating / Authorizing Users

To authenticate your users with GitHub simply call *authorize()* at your login handler:

```
@app.route('/login')
def login():
    return github.authorize()
```

It will redirect the user to GitHub. If the user accepts the authorization request GitHub will redirect the user to your callback URL with the OAuth `code` parameter. Then the extension will make another request to GitHub to obtain access token and call your *authorized_handler()* function with that token. If the authorization fails `oauth_token` parameter will be `None`:

```
@app.route('/github-callback')
@github.authorized_handler
def authorized(oauth_token):
    next_url = request.args.get('next') or url_for('index')
    if oauth_token is None:
        flash("Authorization failed.")
        return redirect(next_url)

    user = User.query.filter_by(github_access_token=oauth_token).first()
    if user is None:
        user = User(oauth_token)
        db_session.add(user)

    user.github_access_token = oauth_token
    db_session.commit()
    return redirect(next_url)
```

Store this token somewhere securely. It is needed later to make requests on behalf of the user.

CHAPTER 4

# Invoking Remote Methods

We need to register a function as a token getter for Github-Flask extension. It will be called automatically by the extension to get the access token of the user. It should return the access token or `None`:

```
@github.access_token_getter
def token_getter():
    user = g.user
    if user is not None:
        return user.github_access_token
```

After setting up you can use the *`get()`*, *`post()`* or other verb methods of the *`GitHub`* object. They will return a dictionary representation of the given API endpoint.

```
@app.route('/repo')
def repo():
    repo_dict = github.get('repos/cenkalti/github-flask')
    return str(repo_dict)
```

CHAPTER 5

# Full Example

A full example can be found in example.py file. Install the required Flask-SQLAlchemy package first. Then edit the file and change `GITHUB_CLIENT_ID` and `GITHUB_CLIENT_SECRET` settings. Then you can run it as a python script:

```
$ pip install Flask-SQLAlchemy
$ python example.py
```

# CHAPTER 6

# API Reference

**class** flask_github.**GitHub**(*app=None*)

Provides decorators for authenticating users with GitHub within a Flask application. Helper methods are also provided interacting with GitHub API.

**access_token_getter**(*f*)

Registers a function as the access_token getter. Must return the access_token used to make requests to GitHub on the user's behalf.

**authorize**(*scope=None*, *redirect_uri=None*, *state=None*)

Redirect to GitHub and request access to a user's data.

**Parameters**

- **scope** (*str*) – List of Scopes for which to request access, formatted as a string or comma delimited list of scopes as a string. Defaults to `None`, resulting in granting read-only access to public information (includes public user profile info, public repository info, and gists). For more information on this, see the examples in presented in the GitHub API Scopes documentation, or see the examples provided below.
- **redirect_uri** (*str*) – Redirect URL to which to redirect the user after authentication. Defaults to `None`, resulting in using the default redirect URL for the OAuth application as defined in GitHub. This URL can differ from the callback URL defined in your GitHub application, however it must be a subdirectory of the specified callback URL, otherwise raises a *GitHubError*. For more information on this, see the examples in presented in the GitHub API Redirect URL documentation, or see the example provided below.
- **state** (*str*) – An unguessable random string. It is used to protect against cross-site request forgery attacks.

For example, if we wanted to use this method to get read/write access to user profile information, in addition to read-write access to code, commit status, etc., we would need to use the Scopes `user` and `repo` when calling this method.

```
github.authorize(scope="user,repo")
```

Additionally, if we wanted to specify a different redirect URL following authorization.

```
# Our application's callback URL is "http://example.com/callback"
redirect_uri="http://example.com/callback/my/path"

github.authorize(scope="user,repo", redirect_uri=redirect_uri)
```

**authorized_handler**(*f*)
Decorator for the route that is used as the callback for authorizing with GitHub. This callback URL can be set in the settings for the app or passed in during authorization.

**get**(*resource*, *params=None*, ***kwargs*)
Shortcut for `request('GET', resource)`.

**post**(*resource*, *data=None*, ***kwargs*)
Shortcut for `request('POST', resource)`. Use this to make POST request since it will also encode `data` to 'application/json' format.

**raw_request**(*method*, *resource*, *access_token=None*, ***kwargs*)
Makes a HTTP request and returns the raw `Response` object.

**request**(*method*, *resource*, *all_pages=False*, ***kwargs*)
Makes a request to the given endpoint. Keyword arguments are passed to the `request()` method. If the content type of the response is JSON, it will be decoded automatically and a dictionary will be returned. Otherwise the `Response` object is returned.

**class** flask_github.**GitHubError**
Raised if a request fails to the GitHub API.

**response**
The `Response` object for the request.

# Python Module Index

## f

# Index