ReplOAuth – A self-replicating app

Technical Specification

ReplOAuth is a python app that uses the lightweight flask web framework. It is deployed and hosted at heroku (https://replicate-myself.herokuapp.com) and is registered as an OAuth app at the github of the developer (refer to the installation document). OAuth App uses GitHub as an identity provider to authenticate as the user who grants access to the app. This document briefly explains the workflow of the app, alongside short descriptions and links to the technologies used and the way they are working together. As a complimentary of this document, the code (webapp.py) contains different remarks that explain different parts of the code in more detail.

Creating the app object and starting the home page

Opening the first page in the browser triggers the `@app.route('/')` in the main code `webapp.py` (will be referred to as *the code* hereafter). This is an application of decorators in python which is heavily used by flask. Decorators are functions that take another function and extend their behavior without explicitly modifying it. All functions that are decorated with a `@app.route()`are triggered by a routing request. `app` is a flask object that is created at Step 3 in *the code*. *The code* then renders the home page and starts the app.

Login process

Pressing the login link on top right of the home page starts the github authorization process through the function coming after `@app.route('login')` in *the code*. it uses our OAuth object (github) to log in and authorize a user by first calling the url for the function `authorized()`. This is a two-step process which uses OAuth2 technology to give access to the app for controlling user's public repo. The request for this access is added to the code at Step 4 by adding `{'scope':'public_repo'}` to the oauth object specifications. The second step is an added warning to the user that they are giving read/write access of their public repository (Figure 1.)

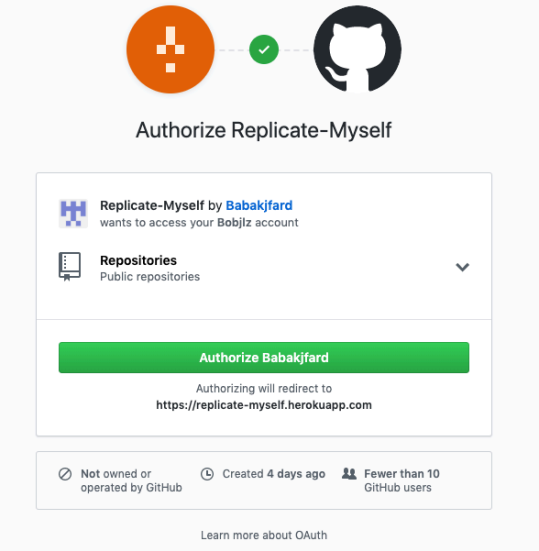

Figure 1. Warning from Github to the user who are giving read/write access to an app

After clicking Authorize button, Github calls the **Authorization callback URL** registered by the app (see the installation document)(here it is https://replicate-myself.herokuapp.com/login/authorize). That triggers the function decorated by `@app.route('/login/authorized')` in *the code*.

Authorized space of the app and getting the session variables

Inside `authorized()` function the code checks if the authorizing has been successful and a response is created by github. Then it stores the `access_token` and `user_data` in the session object. **session** allows you to store information specific to a user from one request to the next. This is implemented on top of cookies for you and signs the cookies cryptographically. What this means is that the user could look at the contents of your cookie but not modify it, unless they know the secret key used for signing. (Setting the secret keys is explained in the Installation Document).

Successful login

If login is successful, the app is redirected to the home page with the variable `logged_in` set to `True`. This variable assignment has happened in Step 6 in the code, using a context processor. Context processor is a function that returns a dictionary which is merged to all templates in an app(here the dictionary is `{'logged_in':True}`). Flask by default uses Jinja2 template engine to maintain the control of its templates (e.g. html files). Checking for the `logged_in` status in `navbar.html` adds the user information on the right top of the page. also home.html adds a form for creating the repo. All these are inside `{% if logged_in %} {% endif %}` block in the mentioned files. The Htmls also use `{% extend`, `{% include`, and `{% block` to do template

inheritance, include a template inside another one, and name different parts of the templates, respectively. More about the syntax of jinja2 can be found here.

Creating the repo and copying the files

After successful authentication of the app by user's github, the user can give a name and push the *Replicate* button, which triggers the function decorated by `@app.route('/replicate')` in the code. The code creates a Githhub object with the access_token saved in the session object. This object is then used to create the repo and the files inside it, using the methods available in Github object for these purposes. The name of the repository is retrieved from the form by request.form of Flask that looks for the input values from the part of the form in the html with `name='repo'`.