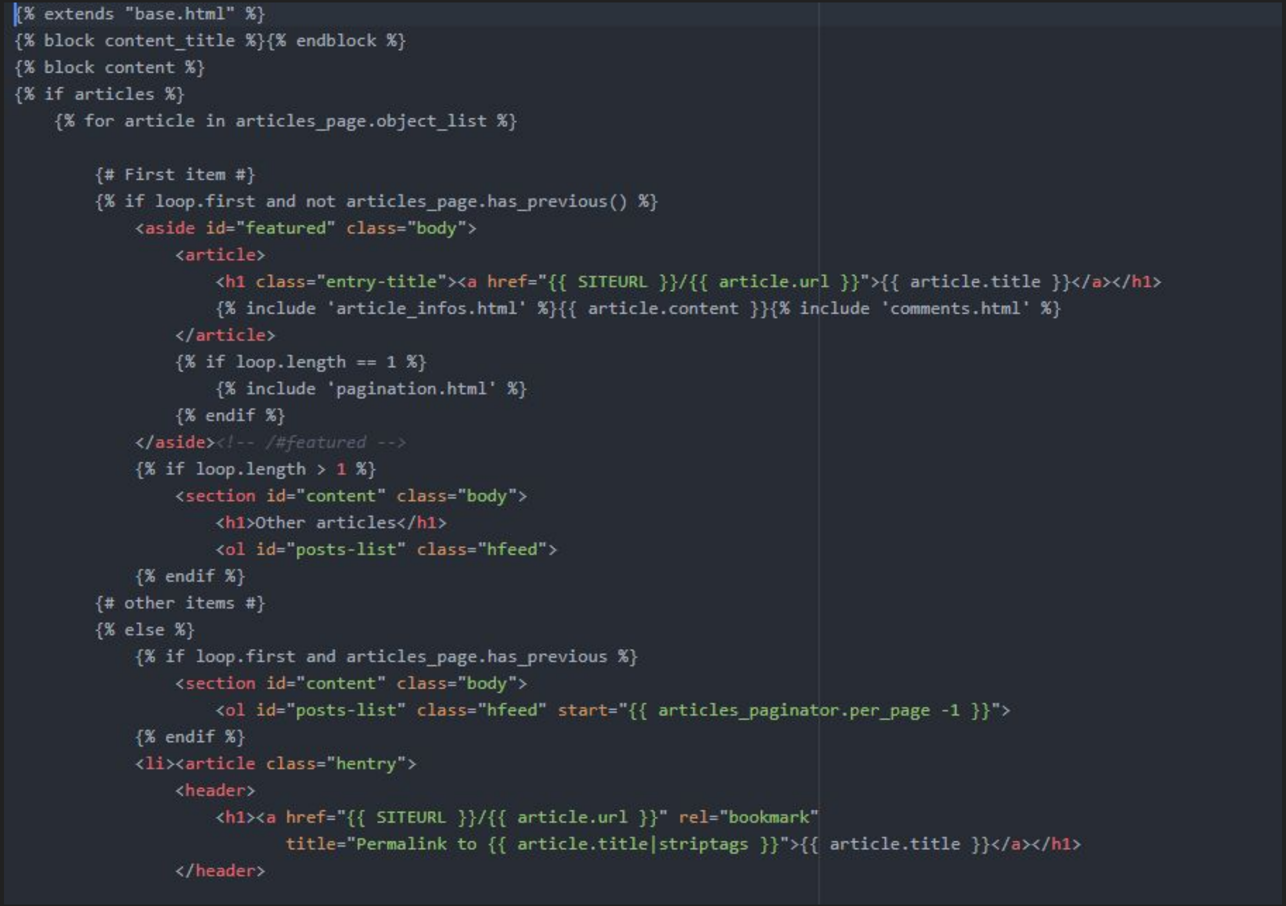
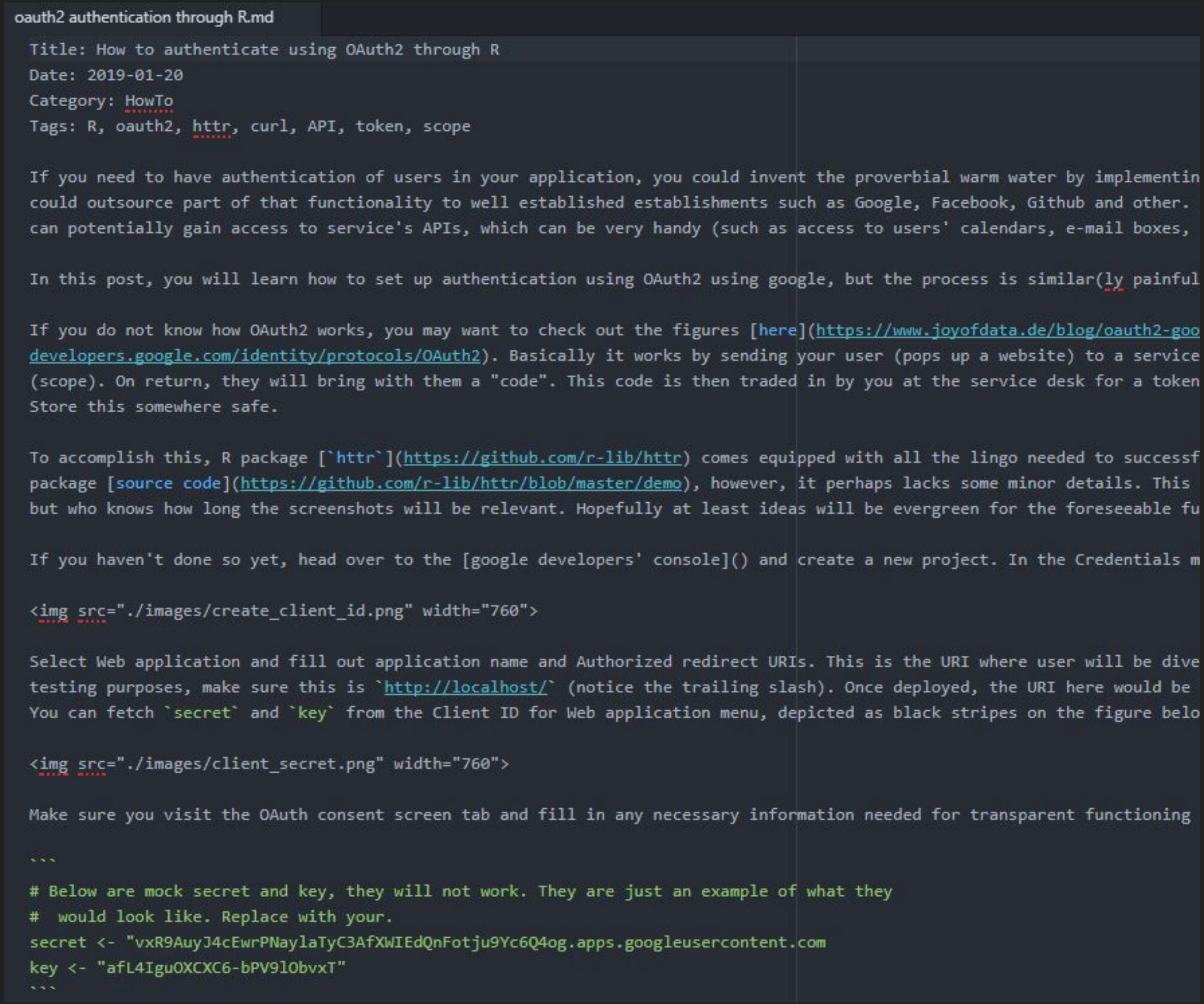
```
import ljpymeetup
import lightningtalk
import datetime

title = 'Static websites are (still) \
         the rage - Pelican static page \
         generator'

company = 'Genialis'
email = 'roman.lustrik@biolitika.si'
social_media_handle = 'romunov'
datum = datetime.datetime(2019, 3, 19)
location = {'name': 'poligon (Tobačna)',
            'xy': [46.0483143, 14.4937637]}
```

```
│
├───content
│   ├───images
│   └───pages
├───output
│   ├───author
│   ├───category
│   ├───drafts
│   ├───feeds
│   ├───images
│   ├───tag
│   └───theme
│       ├───css
│       └───images
│           └───icons
├───raw_themes
│   └───pelican-blueidea
│       ├───static
│       │   ├───css
│       │   └───images
│       │       └───icons
│       └───templates
└───__pycache__
```

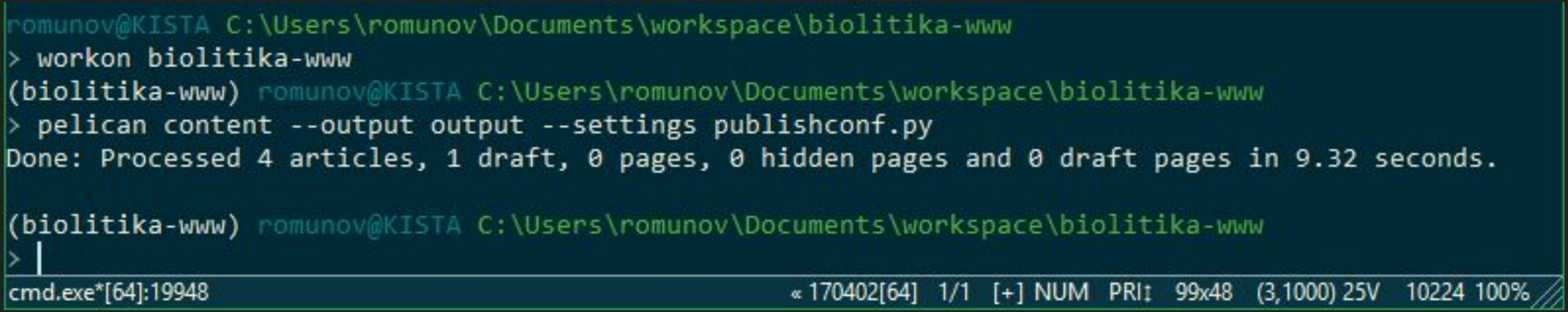


```
{% extends "base.html" %}
{% block content_title %}{% endblock %}
{% block content %}
{% if articles %}
    {% for article in articles_page.object_list %}

        {# First item #}
        {% if loop.first and not articles_page.has_previous() %}
            <aside id="featured" class="body">
                <article>
                    <h1 class="entry-title"><a href="{{ SITEURL }}/{{ article.url }}">{{ article.title }}</a></h1>
                    {% include 'article_infos.html' %}{{ article.content }}{% include 'comments.html' %}
                </article>
                {% if loop.length == 1 %}
                    {% include 'pagination.html' %}
                {% endif %}
            </aside><!-- /#featured -->
            {% if loop.length > 1 %}
                <section id="content" class="body">
                    <h1>Other articles</h1>
                    <ol id="posts-list" class="hfeed">
            {% endif %}
        {# other items #}
        {% else %}
            {% if loop.first and articles_page.has_previous %}
                <section id="content" class="body">
                    <ol id="posts-list" class="hfeed" start="{{ articles_paginator.per_page -1 }}">
            {% endif %}
            <li><article class="hentry">
                <header>
                    <h1><a href="{{ SITEURL }}/{{ article.url }}" rel="bookmark"
                           title="Permalink to {{ article.title|striptags }}">{{ article.title }}</a></h1>
                </header>
```

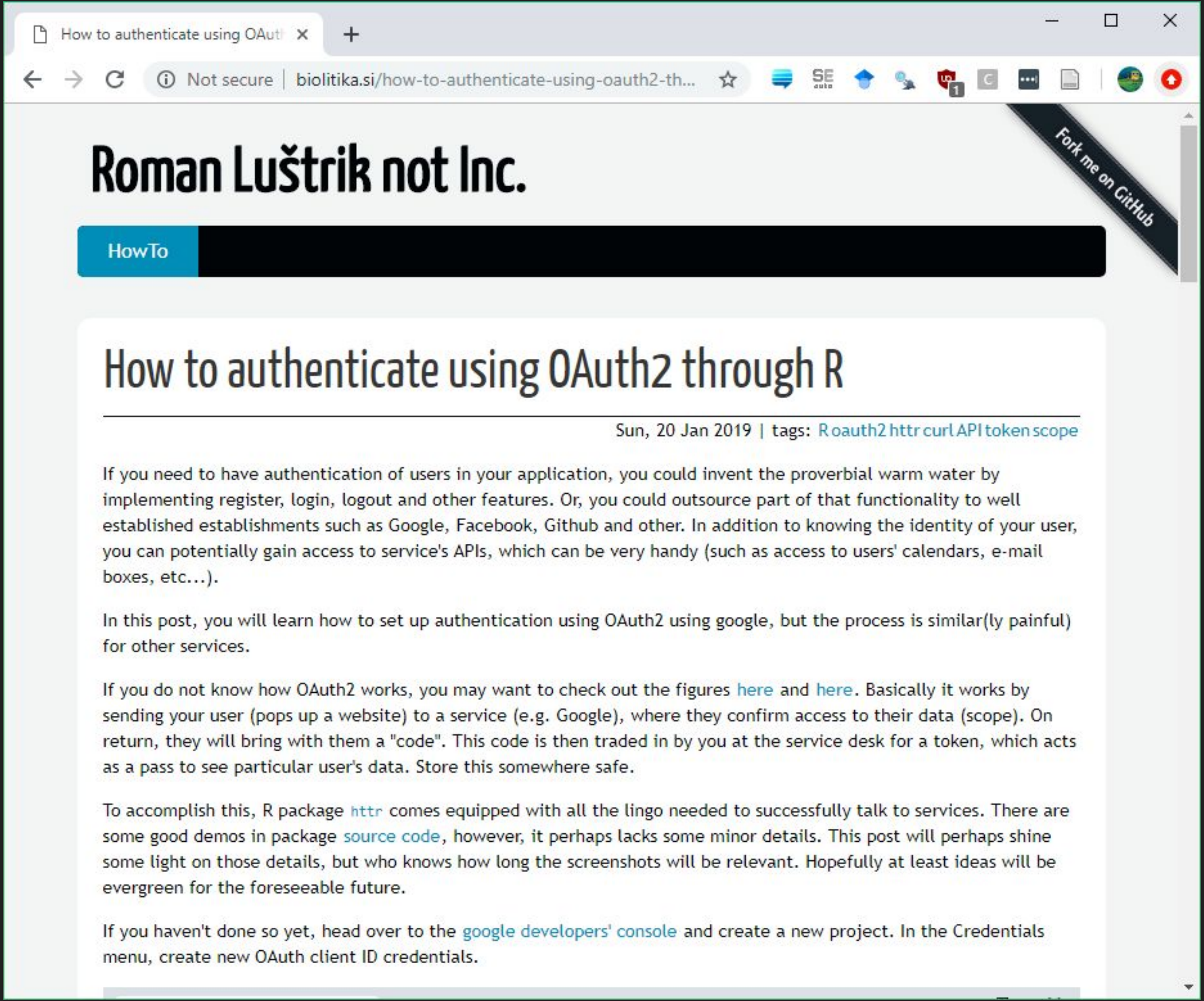


oauth2 authentication through R.md

~~~
Title: How to authenticate using OAuth2 through R
Date: 2019-01-20
Category: HowTo
Tags: R, oauth2, httr, curl, API, token, scope

If you need to have authentication of users in your application, you could invent the proverbial warm water by implementin
could outsource part of that functionality to well established establishments such as Google, Facebook, Github and other.
can potentially gain access to service's APIs, which can be very handy (such as access to users' calendars, e-mail boxes,

In this post, you will learn how to set up authentication using OAuth2 using google, but the process is similar(ly painful

If you do not know how OAuth2 works, you may want to check out the figures [here](https://www.joyofdata.de/blog/oauth2-goo
developers.google.com/identity/protocols/OAuth2). Basically it works by sending your user (pops up a website) to a service
(scope). On return, they will bring with them a "code". This code is then traded in by you at the service desk for a token
Store this somewhere safe.

To accomplish this, R package [`httr`](https://github.com/r-lib/httr) comes equipped with all the lingo needed to successf
package [source code](https://github.com/r-lib/httr/blob/master/demo), however, it perhaps lacks some minor details. This
but who knows how long the screenshots will be relevant. Hopefully at least ideas will be evergreen for the foreseeable fu

If you haven't done so yet, head over to the [google developers' console]() and create a new project. In the Credentials m

<img src="./images/create_client_id.png" width="760">

Select Web application and fill out application name and Authorized redirect URIs. This is the URI where user will be dive
testing purposes, make sure this is `http://localhost/` (notice the trailing slash). Once deployed, the URI here would be
You can fetch `secret` and `key` from the Client ID for Web application menu, depicted as black stripes on the figure belo

<img src="./images/client_secret.png" width="760">

Make sure you visit the OAuth consent screen tab and fill in any necessary information needed for transparent functioning

```
# Below are mock secret and key, they will not work. They are just an example of what they
#  would look like. Replace with your.
secret <- "vxR9AuyJ4cEwrPNaylaTyC3AfXWIEdQnFotju9Yc6Q4og.apps.googleusercontent.com
key <- "afL4IguOXCXC6-bPV9lObvxT"
```
~~~



```
romunov@KISTA C:\Users\romunov\Documents\workspace\biolitika-www
> workon biolitika-www
(biolitika-www) romunov@KISTA C:\Users\romunov\Documents\workspace\biolitika-www
> pelican content --output output --settings publishconf.py
Done: Processed 4 articles, 1 draft, 0 pages, 0 hidden pages and 0 draft pages in 9.32 seconds.

(biolitika-www) romunov@KISTA C:\Users\romunov\Documents\workspace\biolitika-www
> 
cmd.exe*[64]:19948                           « 170402[64]  1/1  [+] NUM  PRI↕  99x48  (3,1000) 25V  10224 100%
```


How to authenticate using OAuth
Not secure
biolitika.si/how-to-authenticate-using-oauth2-th...
Fork me on GitHub
HowTo

Roman Luštrik not Inc.

How to authenticate using OAuth2 through R

Sun, 20 Jan 2019 | tags: R oauth2 httr curl API token scope

If you need to have authentication of users in your application, you could invent the proverbial warm water by implementing register, login, logout and other features. Or, you could outsource part of that functionality to well established establishments such as Google, Facebook, Github and other. In addition to knowing the identity of your user, you can potentially gain access to service's APIs, which can be very handy (such as access to users' calendars, e-mail boxes, etc...).

In this post, you will learn how to set up authentication using OAuth2 using google, but the process is similar(ly painful) for other services.

If you do not know how OAuth2 works, you may want to check out the figures here and here. Basically it works by sending your user (pops up a website) to a service (e.g. Google), where they confirm access to their data (scope). On return, they will bring with them a "code". This code is then traded in by you at the service desk for a token, which acts as a pass to see particular user's data. Store this somewhere safe.

To accomplish this, R package `httr` comes equipped with all the lingo needed to successfully talk to services. There are some good demos in package source code, however, it perhaps lacks some minor details. This post will perhaps shine some light on those details, but who knows how long the screenshots will be relevant. Hopefully at least ideas will be evergreen for the foreseeable future.

If you haven't done so yet, head over to the google developers' console and create a new project. In the Credentials menu, create new OAuth client ID credentials.

@romunov

roman.lustrik@biolitika.si