

Forum:[Hack Forums](#)

Thread: [Python Wordpress Class](#)

MiguelSanchez - January 26, 2022, 5:14 am

Basically I had wanted to connect to wordpress and do some site updates for clients from directly within python, I couldn't find any wordpress library that wasn't a hobby project on someone's github.

I found [robingeuens\(dot\)com/blog/python-wordpress-api](https://robingeuens.com/blog/python-wordpress-api) which had everything I needed to know. Copying that code and tweaking everything to place it within a single class I streamlined authentication and wrote a few helper functions. Things like "page_id" & post_id which can return an elements id by searching for a link, title, or slug (soon these will all be regular expressions, at the moment they're just literal case insensitive searches similar to your browser's find function)

I probably could've condensed it by using higher order functions to perform both the posts and the pages within one unified method. I probably will on future versions?

Code import requests

import base64

class Wordpress(target):

def __init__(self, domain, user=None, admin=None, wpjson=None):

if wpjson==None:

wpjson=domain+"/wp-json/"

#check wpjson

#wpjson=domain+"index.php/wp-json"

self.wpjson=wpjson

self.domain=domain

self.user=user

self.admin=admin

self.wpadmin=domain+"/wp-admin/"

self.session=None

self.wplogin=domain+"/wp-login.php"

self.gentoken()

self.posts=None

self.pages=None

def displaypages(self):

if self.pages==None:

self.getpages()

for i in self.pages:

print("Title: %s\nLink: %s \nID: %s"%(i['title']['rendered'],i['link'],i['id']))

def displayposts(self):

if self.posts==None:

self.getposts()

for i in self.posts:

print("Title: %s\nLink: %s \nID: %s"%(i['title']['rendered'],i['link'],i['id']))

def gentoken(self):

wordpress_credentials =self.user["username"]+":"+self.user["password"]

self.token = base64.b64encode(wordpress_credentials.encode())

self.wordpress_header = {'Authorization': 'Basic ' + self.token.decode('utf-8')}

Code def getposts(self):

```

api_url = self.wpjson+'wp/v2/posts'
#response = requests.get(api_url)
if self.session==None:
    self.login()
response = self.session.get(api_url)
response_header=response.headers
response_json = response.json()
self.posts=response_json
return response_header,response_json
def getpages(self):
    api_url = self.wpjson+'wp/v2/pages'
    #response = requests.get(api_url)
    if self.session==None:
        self.login()
    response = self.session.get(api_url)
    response_header=response.headers
    response_json = response.json()
    self.pages=response_json
    return response_header,response_json
def login(self,redirect=None):
    if redirect==None:
        redirect=self.admin
    with requests.Session() as s:
        #self.headers = {'Cookie':"test_cookie=WP Cookie check"}
        data={'log':self.user["username"],
            'pwd':self.user["password"],
            'wp-submit':'Log In',
            'redirect_to':redirect}
        s.post(self.wplogin,headers=self.wordpress_header,data=data)
        self.session=s
        #resp=s.get(self.wpadmin)
def create_post(self,content,title="API TEST",slug="test-post"):
    api_url = self.wpjson+'/wp/v2/posts'
    postdata = {
        'title' : title,
        'status': 'publish',
        'slug' : slug,
        'content': content
    }
    response = self.session.post(api_url,headers=self.wordpress_header, json=postdata)
    print("creating post %s"%title)
    if response.status_code=='201':
        print('success')
    else:
        print('Failed status code %s'%response.status_code)
    return response
def create_page(self,content,title="API TEST",slug="test-page"):
    api_url = self.wpjson+'/wp/v2/pages'
    postdata = {
        'title' : title,
        'status': 'publish',
        'slug' : slug,
        'content': content
    }

```

```

    }
    response = self.session.post(api_url,headers=self.wordpress_header, json=postdata)
    print("creating post %s"%title)
    if response.status_code==201:
        print('success')
    else:
        print('Failed!\n Status Code:%s'%response.status_code)
    return response
def page_id(self,title=None,slug=None,link=None):
    if self.pages==None:
        self.getpages()
    if title==None and link==None and slug==None:
        print("Needs title, slug, or link to lookup ID")
    for i in self.pages:
        if title:
            if i['title']['rendered'].lower()==title.lower():
                return i['id']
        if slug:
            if i['slug'].lower()==slug.lower():
                return i['id']
        if link:
            if i['link'].lower()==link.lower():
                return i['id']
    return None
def post_id(self,title=None,slug=None,link=None):
    if self.posts==None:
        self.getposts()
    if title==None and link==None and slug==None:
        print("Needs title, slug, or link to lookup ID")
    for i in self.posts:
        if i['title']['rendered'].lower()==title.lower():
            return i['id']
        if i['slug'].lower()==slug.lower():
            return i['id']
        if i['link'].lower()==link.lower():
            return i['id']
    return None
def update_post(self,post_id,content,title,slug):
    api_url = self.wpjson+'/wp/v2/posts/'
    data = {
        'title' : title,
        'status': 'publish',
        'slug' : slug,
        'content': content
    }
    response = requests.post(api_url+str(post_id),headers=self.wordpress_header, json=data)
    print("editing post id:%s"%str(post_id))
    if response.status_code==200 or response.status_code==201:
        print('success')
    else:
        print('Failed!\n Status Code:%s'%response.status_code)
    return response
def update_page(self,page_id,content,title,slug):

```

```

api_url = self.wpjson+'/wp/v2/pages/'
data = {
'title' : title,
'status': 'publish',
'slug' : slug,
'content': content
}
response = requests.post(api_url+str(page_id),headers=self.wordpress_header, json=data)
print("editing page id:%s"%str(page_id))
if response.status_code==200 or response.status_code==201:
    print('success')
else:
    print('Failed!\n Status Code:%s'%response.status_code)
return response
def delete_page(self,page_id):
    api_url = self.wpjson+'/wp/v2/pages/'
    response = self.session.delete(api_url + page_id, headers=self.wordpress_header)
    return response
def delete_post(self,post_id):
    api_url = self.wpjson+'/wp/v2/posts/'
    response = self.session.delete(api_url + post_id, headers=self.wordpress_header)
    return response

```

This is the end of the class had to split into two due to file size

Hackers - January 28, 2022, 4:51 am

Unless you can find a plugin that makes it all API-accessible for Python your best bet is probably to just give Python direct access to database and run the script on the server itself so it can modify files. You can run Python as a shell from PHP if you want to do that the easy way, but you'll need to be careful with any user data going into the `exec()` - or similar functions - of PHP which basically expose command line access.

MiguelSanchez - January 28, 2022, 8:03 am

Yeah, I haven't really looked into what wordpress is doing in the database yet, if I was trying to run python within the wordpress I'd definitely just do it via PHP.

This is just part of a python library I'm making for pushing content to directly to multiple end-points (Social media posts, wordpress post, emails, post to an upload form all at once). This is just the Wordpress class from it (since I finished it first) but it's made to be run on a computer other than the server the Wordpress is on.

I haven't really discovered any major limitations to this method yet.

it's used like

```

Code Acc={"username":"Admin","password":"7zPF Bq3y zaJN iYBM nPm0 b9qE"}
#Application password generated in wordpress dashboard
Site=Wordpress("https://mysite.com.au",user=Acc)
Site.update_page(Site.page_id(title='landing page'),content="landing page updated via
API",title='landing page',slug='landing-page-updated')
Site.create_page("CONTENT GO HERE",title="my new page",slug="SLUG")

```

(Jan 28, 2022 05:51 AM)nodes Wrote: [CITE](#) best bet is probably to just give Python direct access to database

I'm in a wordpress MYSQL database at the moment, moving some stuff from a client's old site, to the new site I'm building, sadly the old site isn't functional via apache anymore (as in the domain no longer points to its directory)

I didn't plan ahead, should've moved it to a subdomain to keep it active.

But I just realised the wp_posts table has the column post_content stored exactly as you would see it in the front end, to the extent you can literally copy it from the MYsql table, and paste it into the gutenber editor and have it appear as intended.

As this is the first time I've gone into wordpress's database (I usually make my own separate DB instead of using WP's) I had no idea. That's actually super cool!