# -\*- coding: utf-8 -\*-

#

# jQuery File Upload Plugin GAE Python Example 2.0

# https://github.com/blueimp/jQuery-File-Upload

#

# Copyright 2011, Sebastian Tschan

# https://blueimp.net

#

# Licensed under the MIT license:

# http://www.opensource.org/licenses/MIT

#

from \_\_future\_\_ import with\_statement

from google.appengine.api import files, images

from google.appengine.ext import blobstore, deferred

from google.appengine.ext.webapp import blobstore\_handlers

import json, re, urllib, webapp2

WEBSITE = 'http://blueimp.github.com/jQuery-File-Upload/'

MIN\_FILE\_SIZE = 1 # bytes

MAX\_FILE\_SIZE = 5000000 # bytes

IMAGE\_TYPES = re.compile('image/(gif|p?jpeg|(x-)?png)')

ACCEPT\_FILE\_TYPES = IMAGE\_TYPES

THUMBNAIL\_MODIFICATOR = '=s80' # max width / height

EXPIRATION\_TIME = 300 # seconds

def cleanup(blob\_keys):

blobstore.delete(blob\_keys)

class UploadHandler(webapp2.RequestHandler):

def initialize(self, request, response):

super(UploadHandler, self).initialize(request, response)

self.response.headers['Access-Control-Allow-Origin'] = '\*'

self.response.headers[

'Access-Control-Allow-Methods'

] = 'OPTIONS, HEAD, GET, POST, PUT, DELETE'

def validate(self, file):

if file['size'] < MIN\_FILE\_SIZE:

file['error'] = 'File is too small'

elif file['size'] > MAX\_FILE\_SIZE:

file['error'] = 'File is too big'

elif not ACCEPT\_FILE\_TYPES.match(file['type']):

file['error'] = 'Filetype not allowed'

else:

return True

return False

def get\_file\_size(self, file):

file.seek(0, 2) # Seek to the end of the file

size = file.tell() # Get the position of EOF

file.seek(0) # Reset the file position to the beginning

return size

def write\_blob(self, data, info):

blob = files.blobstore.create(

mime\_type=info['type'],

\_blobinfo\_uploaded\_filename=info['name']

)

with files.open(blob, 'a') as f:

f.write(data)

files.finalize(blob)

return files.blobstore.get\_blob\_key(blob)

def handle\_upload(self):

results = []

blob\_keys = []

for name, fieldStorage in self.request.POST.items():

if type(fieldStorage) is unicode:

continue

result = {}

result['name'] = re.sub(r'^.\*\\', '',

fieldStorage.filename)

result['type'] = fieldStorage.type

result['size'] = self.get\_file\_size(fieldStorage.file)

if self.validate(result):

blob\_key = str(

self.write\_blob(fieldStorage.value, result)

)

blob\_keys.append(blob\_key)

result['delete\_type'] = 'DELETE'

result['delete\_url'] = self.request.host\_url +\

'/?key=' + urllib.quote(blob\_key, '')

if (IMAGE\_TYPES.match(result['type'])):

try:

result['url'] = images.get\_serving\_url(

blob\_key,

secure\_url=self.request.host\_url\

.startswith('https')

)

result['thumbnail\_url'] = result['url'] +\

THUMBNAIL\_MODIFICATOR

except: # Could not get an image serving url

pass

if not 'url' in result:

result['url'] = self.request.host\_url +\

'/' + blob\_key + '/' + urllib.quote(

result['name'].encode('utf-8'), '')

results.append(result)

deferred.defer(

cleanup,

blob\_keys,

\_countdown=EXPIRATION\_TIME

)

return results

def options(self):

pass

def head(self):

pass

def get(self):

self.redirect(WEBSITE)

def post(self):

if (self.request.get('\_method') == 'DELETE'):

return self.delete()

result = {'files': self.handle\_upload()}

s = json.dumps(result, separators=(',',':'))

redirect = self.request.get('redirect')

if redirect:

return self.redirect(str(

redirect.replace('%s', urllib.quote(s, ''), 1)

))

if 'application/json' in self.request.headers.get('Accept'):

self.response.headers['Content-Type'] = 'application/json'

self.response.write(s)

def delete(self):

blobstore.delete(self.request.get('key') or '')

class DownloadHandler(blobstore\_handlers.BlobstoreDownloadHandler):

def get(self, key, filename):

if not blobstore.get(key):

self.error(404)

else:

# Cache for the expiration time:

self.response.headers['Cache-Control'] =\

'public,max-age=%d' % EXPIRATION\_TIME

self.send\_blob(key, save\_as=filename)

app = webapp2.WSGIApplication(

[

('/', UploadHandler),

('/([^/]+)/([^/]+)', DownloadHandler)

],

debug=True

)