'''

gdrive (Google Drive ) for KODI / XBMC Plugin

Copyright (C) 2013-2016 ddurdle

This program is free software: you can redistribute it and/or modify

it under the terms of the GNU General Public License as published by

the Free Software Foundation, either version 3 of the License, or

(at your option) any later version.

This program is distributed in the hope that it will be useful,

but WITHOUT ANY WARRANTY; without even the implied warranty of

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

GNU General Public License for more details.

You should have received a copy of the GNU General Public License

along with this program. If not, see <http://www.gnu.org/licenses/>.

'''

import os

import re

import urllib, urllib2

import cookielib

import json

import xbmc, xbmcaddon, xbmcgui, xbmcplugin

import authorization

import crashreport

from resources.lib import package

from resources.lib import file

from resources.lib import folder

class gSheets\_api4:

API\_URL = 'https://sheets.googleapis.com/v4/spreadsheets'

def \_\_init\_\_(self, service, addon, user\_agent):

self.addon = addon

self.service = service

# self.crashreport = crashreport

# self.crashreport.sendError('test','test')

self.user\_agent = user\_agent

return

#

# returns a spreadsheet

#

def createSpreadsheet(self):

#header = { 'User-Agent' : self.user\_agent, 'Authorization' : 'GoogleLogin auth=%s' % self.authorization.getToken('wise'), 'GData-Version' : '3.0', 'Content-Type': 'application/atom+xml' }

entry = '{"properties": { "title": "TEST123" }}'

# entry = { 'properties' : {'title': 'TEST1234'}}

url = self.API\_URL #+ '?key=AIzaSyD-a9IF8KKYgoC3cpgS-Al7hLQDbugrDcw&alt=json'

# url = 'https://sheets.googleapis.com/v4/spreadsheets/1lrARPXpjLAO-edm5J9p0UK7nmkukST6bv07u8ai1MY8'

req = urllib2.Request(url, entry, self.service.getHeadersList(isPOST=True))

#req = urllib2.Request(url, json.dumps(entry), self.service.getHeadersList(isPOST=True, isJSON=True))

# req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, entry, self.service.getHeadersList(isPOST=True))

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return False

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return False

response\_data = response.read()

response.close()

return True

#

# append data to spreadsheet

#

def addRows(self, spreadsheetID):

entry = '{"values": [[ "title", "TEST123" ]]}'

url = self.API\_URL + '/'+spreadsheetID+'/values/A1:append?valueInputOption=USER\_ENTERED'#values/Sheet1!A1:A3?valueInputOption=USER\_ENTERED'

# url = 'https://sheets.googleapis.com/v4/spreadsheets/1lrARPXpjLAO-edm5J9p0UK7nmkukST6bv07u8ai1MY8'

req = urllib2.Request(url, entry, self.service.getHeadersList(isPOST=True))

#req = urllib2.Request(url, json.dumps(entry), self.service.getHeadersList(isPOST=True, isJSON=True))

# req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, entry, self.service.getHeadersList(isPOST=True))

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return False

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return False

response\_data = response.read()

response.close()

return True

#

# returns a list of spreadsheets and a link to their worksheets

#

def getSpreadsheetList(self):

url = 'https://spreadsheets.google.com/feeds/spreadsheets/private/full'

spreadsheets = {}

while True:

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.msg != '':

xbmcgui.Dialog().ok(self.addon.getLocalizedString(30000), e.msg)

xbmc.log(self.addon.getAddonInfo('getSpreadsheetList') + ': ' + str(e), xbmc.LOGERROR)

self.crashreport.sendError('getSpreadsheetList',str(e))

else:

if e.msg != '':

xbmcgui.Dialog().ok(self.addon.getLocalizedString(30000), e.msg)

xbmc.log(self.addon.getAddonInfo('getSpreadsheetList') + ': ' + str(e), xbmc.LOGERROR)

self.crashreport.sendError('getSpreadsheetList',str(e))

response\_data = response.read()

response.close()

for r in re.finditer('<title [^\>]+\>([^<]+)</title><content [^\>]+\>[^<]+</content><link rel=\'[^\#]+\#worksheetsfeed\' type=\'application/atom\+xml\' href=\'([^\']+)\'' ,

response\_data, re.DOTALL):

title,url = r.groups()

# must be read/write spreadsheet, skip read-only

regexp = re.compile(r'/private/values')

if regexp.search(url) is None:

spreadsheets[title] = url

nextURL = ''

for r in re.finditer('<link rel=\'next\' type=\'[^\']+\' href=\'([^\']+)\'' ,

response\_data, re.DOTALL):

nextURL = r.groups()

if nextURL == '':

break

else:

url = nextURL[0]

return spreadsheets

#

# returns a list of spreadsheets contained in the Google Docs account

#

def createWorksheet(self,url,title,cols,rows):

header = { 'User-Agent' : self.user\_agent, 'Authorization' : 'GoogleLogin auth=%s' % self.authorization.getToken('wise'), 'GData-Version' : '3.0', 'Content-Type': 'application/atom+xml' }

entry = '<?xml version=\'1.0\' encoding=\'UTF-8\'?><entry xmlns="http://www.w3.org/2005/Atom" xmlns:gs="http://schemas.google.com/spreadsheets/2006"><title>A worksheetdadf</title><gs:rowCount>100</gs:rowCount><gs:colCount>20</gs:colCount></entry>'

req = urllib2.Request(url, entry, header)

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return False

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return False

response\_data = response.read()

response.close()

return True

#

# returns a list of spreadsheets contained in the Google Docs account

#

def createRow(self,url, folderID, folderName, fileID, fileName):

# header = { 'User-Agent' : self.user\_agent, 'Authorization' : 'GoogleLogin auth=%s' % self.authorization.getToken('wise'), 'GData-Version' : '3.0', 'Content-Type': 'application/atom+xml'}

header = { 'User-Agent' : self.user\_agent, 'Authorization' : 'Bearer ' + self.service.authorization.getToken('auth\_access\_token'), 'GData-Version' : '3.0', 'Content-Type': 'application/atom+xml'}

entry = '<?xml version=\'1.0\' encoding=\'UTF-8\'?><entry xmlns="http://www.w3.org/2005/Atom" xmlns:gsx="http://schemas.google.com/spreadsheets/2006/extended"> <gsx:source>S3E12 - The Red Dot.avi-0002</gsx:source><gsx:nfo>test.nfo</gsx:nfo><gsx:show>Seinfeld</gsx:show><gsx:season>3</gsx:season><gsx:episode>1</gsx:episode><gsx:part>1</gsx:part><gsx:watched>0</gsx:watched><gsx:duration>1</gsx:duration></entry>'

req = urllib2.Request(url, entry, header)

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return False

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return False

response\_data = response.read()

response.close()

return True

#

# returns a list of spreadsheets contained in the Google Docs account

#

def createMediaStatus(self, url, package, resume='', watched='', updated=''):

import time

updated = time.strftime("%Y%m%d%H%M")

# header = { 'User-Agent' : self.user\_agent, 'Authorization' : 'GoogleLogin auth=%s' % self.authorization.getToken('wise'), 'GData-Version' : '3.0', 'Content-Type': 'application/atom+xml'}

header = { 'User-Agent' : self.user\_agent, 'Authorization' : 'Bearer ' + self.service.authorization.getToken('auth\_access\_token'), 'Content-Type': 'application/atom+xml'}

#entry = '<?xml version=\'1.0\' encoding=\'UTF-8\'?><entry xmlns="http://www.w3.org/2005/Atom" xmlns:gsx="http://schemas.google.com/spreadsheets/2006/extended"> <gsx:folderid>'+str(package.folder.id)+'</gsx:folderid><gsx:foldername>'+str(package.folder.title)+'</gsx:foldername><gsx:fileid>'+str(package.file.id)+'</gsx:fileid><gsx:filename>'+str(package.file.title)+'</gsx:filename><gsx:nfo></gsx:nfo><gsx:order></gsx:order><gsx:watched>'+str(watched)+'</gsx:watched><gsx:resume>'+str(resume)+'</gsx:resume><gsx:updated>'+str(updated)+'</gsx:updated></entry>'

## entry = '<?xml version=\'1.0\' encoding=\'UTF-8\'?><entry xmlns="http://www.w3.org/2005/Atom" xmlns:gsx="http://schemas.google.com/spreadsheets/2006/extended"><gsx:fileid>'+str(package.file.id)+'</gsx:fileid><gsx:filename>'+str(package.file.title)+'</gsx:filename><gsx:watched>'+str(watched)+'</gsx:watched><gsx:resume>'+str(resume)+'</gsx:resume>'

# entry = '<?xml version=\'1.0\' encoding=\'UTF-8\'?><entry xmlns="http://www.w3.org/2005/Atom" xmlns:gs="http://schemas.google.com/spreadsheets/2006" xmlns:gsx="http://schemas.google.com/spreadsheets/2006/extended"><entry>' + '<gsx:fileid>'+str(package.file.id)+'</gsx:fileid><gsx:filename>'+str(package.file.title)+'</gsx:filename><gsx:watched>'+str(watched)+'</gsx:watched><gsx:resume>'+str(resume)+'</gsx:resume></entry>'

# entry = '<entry xmlns="http://www.w3.org/2005/Atom" xmlns:gsx="http://schemas.google.com/spreadsheets/2006/extended"><entry>' + '<gsx:fileid>'+str(package.file.id)+'</gsx:fileid><gsx:filename>'+str(package.file.title)+'</gsx:filename><gsx:watched>'+str(watched)+'</gsx:watched><gsx:resume>'+str(resume)+'</gsx:resume></entry>'

entry = '<entry xmlns="http://www.w3.org/2005/Atom" xmlns:gsx="http://schemas.google.com/spreadsheets/2006/extended">' + '<gsx:foldername>'+str(package.folder.title)+'</gsx:foldername><gsx:folderid>'+str(package.folder.id)+'</gsx:folderid><gsx:fileid>'+str(package.file.id)+'</gsx:fileid><gsx:filename>'+str(package.file.title)+'</gsx:filename><gsx:watched>'+str(watched)+'</gsx:watched><gsx:md5>'+str(package.file.checksum)+'</gsx:md5><gsx:resume>'+str(resume)+'</gsx:resume><gsx:updated>'+str(updated)+'</gsx:updated></entry>'

req = urllib2.Request(url, entry, header)

# req = urllib2.Request(url, entry, self.service.getHeadersList(isPOST=True))

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

header = { 'User-Agent' : self.user\_agent, 'Authorization' : 'Bearer ' + self.service.authorization.getToken('auth\_access\_token'), 'Content-Type': 'application/atom+xml'}

req = urllib2.Request(url, entry, header)

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return False

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return False

response\_data = response.read()

response.close()

return True

#

# returns a list of spreadsheets contained in the Google Docs account

#

def createHeaderRow(self,url):

header = { 'User-Agent' : self.user\_agent, 'Authorization' : 'GoogleLogin auth=%s' % self.authorization.getToken('wise'), 'GData-Version' : '3.0', "If-Match" : '\*', 'Content-Type': 'application/atom+xml'}

entry = '<?xml version=\'1.0\' encoding=\'UTF-8\'?><entry xmlns="http://www.w3.org/2005/Atom" xmlns:gsx="http://schemas.google.com/spreadsheets/2006/extended"> <gsx:hours>1</gsx:hours></entry>'

req = urllib2.Request(url, entry, header)

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

header = { 'User-Agent' : self.user\_agent, 'Authorization' : 'GoogleLogin auth=%s' % self.authorization.getToken('wise'), 'GData-Version' : '3.0', "If-Match" : '\*', 'Content-Type': 'application/atom+xml'}

req = urllib2.Request(url, entry, header)

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return False

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return False

response\_data = response.read()

response.close()

return True

#

# returns a list of worksheets with a link to their listfeeds

#

def getSpreadsheetWorksheets(self,url):

worksheets = {}

while True:

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('getSpreadsheetWorksheets') + ': ' + str(e), xbmc.LOGERROR)

else:

xbmc.log(self.addon.getAddonInfo('getSpreadsheetWorksheets') + ': ' + str(e), xbmc.LOGERROR)

response\_data = response.read()

response.close()

for r in re.finditer('<title[^>]+\>([^<]+)</title><content[^>]+\>[^<]+</content><link rel=\'[^\#]+\#listfeed\' type=\'application/atom\+xml\' href=\'([^\']+)\'' ,

response\_data, re.DOTALL):

title,url = r.groups()

worksheets[title] = url

nextURL = ''

for r in re.finditer('<link rel=\'next\' type=\'[^\']+\' href=\'([^\']+)\'' ,

response\_data, re.DOTALL):

nextURL = r.groups()

if nextURL == '':

break

else:

url = nextURL[0]

return worksheets

#

# returns a list of worksheets with a link to their listfeeds

#

def getSpreadsheetWorksheets(self,url):

worksheets = {}

while True:

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('getSpreadsheetWorksheets') + ': ' + str(e), xbmc.LOGERROR)

else:

xbmc.log(self.addon.getAddonInfo('getSpreadsheetWorksheets') + ': ' + str(e), xbmc.LOGERROR)

response\_data = response.read()

response.close()

for r in re.finditer('<title[^>]+\>([^<]+)</title><content[^>]+\>[^<]+</content><link rel=\'[^\#]+\#listfeed\' type=\'application/atom\+xml\' href=\'([^\']+)\'' ,

response\_data, re.DOTALL):

title,url = r.groups()

worksheets[title] = url

nextURL = ''

for r in re.finditer('<link rel=\'next\' type=\'[^\']+\' href=\'([^\']+)\'' ,

response\_data, re.DOTALL):

nextURL = r.groups()

if nextURL == '':

break

else:

url = nextURL[0]

return worksheets

#spreadsheet STRM

def getSTRMplaybackMovie(self,url,title,year):

params = urllib.urlencode({'title': '"' +str(title)+'"'}, {'year': year})

url = url + '?sq=' + params

files = {}

while True:

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return ''

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return ''

response\_data = response.read()

count=0;

for r in re.finditer('<entry>.\*?<gsx:part>([^<]\*)</gsx:part><gsx:mins>([^<]\*)</gsx:mins><gsx:resolution>([^<]\*)</gsx:resolution><gsx:version>([^<]\*)</gsx:version>.\*?<gsx:fileid>([^<]\*)</gsx:fileid></entry>' ,

response\_data, re.DOTALL):

files[count] = r.groups()

#source,nfo,show,season,episode,part,watched,duration

#channel,month,day,weekday,hour,minute,show,order,includeWatched

count = count + 1

nextURL = ''

for r in re.finditer('<link rel=\'next\' type=\'[^\']+\' href=\'([^\']+)\'' ,

response\_data, re.DOTALL):

nextURL = r.groups()

response.close()

if nextURL == '':

break

else:

url = nextURL[0]

if len(files) == 0:

return ''

elif len(files) == 1:

return files[0][4]

options = []

if files:

for item in files:

option = ''

if str(files[item][0]) != '':

option = option + 'part '+ str(files[item][0])+ ' - '

option = option + 'resolution ' + str(files[item][2]) + 'p - mins ' + str(files[item][1])

if str(files[item][3]) != '':

option = option + ' - version ' + str(files[item][3])

options.append( option )

ret = xbmcgui.Dialog().select(self.addon.getLocalizedString(30112), options)

return files[ret][4]

#spreadsheet STRM

def getMovies(self, url, genre=None, year=None, title=None, country=None, director=None, studio=None):

# params = urllib.urlencode({'title': '"' +str(title)+'"'}, {'year': year})

#url = url + '?sq=' + params

# url = url + 'gviz/?tq=SELECT%20B%2C%20C%2C%20D%2C%20F%20WHERE%20C%20CONTAINS%20%27Florida%27'

for r in re.finditer('list/([^\/]+)\/' ,

url, re.DOTALL):

spreadsheetID = r.group(1)

url = 'https://docs.google.com/spreadsheets/d/'+spreadsheetID+'/gviz/tq?tqx=out.csv'

#all genre

#url = url + '&tq=select%20D%2Ccount(A)%20group%20by%20D'

if year is not None:

#exclude multiple genre

url = url + '&tq=select%20A%2CB%2CC%2CD%2CE%2CF%2CG%2CH%2CI%2CJ%2CK%20where%20B%20%3D%20'+str(year)

elif genre is not None:

#exclude multiple genre

url = url + '&tq=select%20A%2CB%2CC%2CD%2CE%2CF%2CG%2CH%2CI%2CJ%2CK%20where%20D%20contains%20\''+str(genre)+'\''

elif country is not None:

#exclude multiple genre

country = re.sub(' ', '%20', country)

# url = url + '&tq=select%20\*%20where%20H%20%3D%20\''+str(country)+'\''

url = url + '&tq=select%20A%2CB%2CC%2CD%2CE%2CF%2CG%2CH%2CI%2CJ%2CK%20where%20H%20%3D%20\''+str(country)+'\''

elif director is not None:

#exclude multiple genre

director = re.sub(' ', '%20', director)

url = url + '&tq=select%20A%2CB%2CC%2CD%2CE%2CF%2CG%2CH%2CI%2CJ%2CK%20where%20J%20%3D%20\''+str(director)+'\''

elif studio is not None:

#exclude multiple genre

studio = re.sub(' ', '%20', studio)

url = url + '&tq=select%20A%2CB%2CC%2CD%2CE%2CF%2CG%2CH%2CI%2CJ%2CK%20where%20I%20%3D%20\''+str(studio)+'\''

elif title is not None and title != '#all':

#title star with A

url = url + '&tq=select%20A%2CB%2CC%2CD%2CE%2CF%2CG%2CH%2CI%2CJ%2CK%20where%20A%20starts%20with%20\''+str(title).lower()+ '\'%20or%20A%20starts%20with%20\''+str(title).upper()+'\''

elif title is not None and title == '#all':

#title star with A

url = url + '&tq=select%20A%2CB%2CC%2CD%2CE%2CF%2CG%2CH%2CI%2CJ%2CK'#%20where%20A%20starts%20with%20\'A\'%20or%20A%20starts%20with%20\'a\'%20or%20A%20starts%20with%20\'B\'%20or%20A%20starts%20with%20\'b\'%20or%20A%20starts%20with%20\'C\'%20or%20A%20starts%20with%20\'c\'%20or%20A%20starts%20with%20\'D\'%20or%20A%20starts%20with%20\'d\'%20or%20A%20starts%20with%20\'E\'%20or%20A%20starts%20with%20\'e\'%20or%20A%20starts%20with%20\'F\'%20or%20A%20starts%20with%20\'f\'%20or%20A%20starts%20with%20\'G\'%20or%20A%20starts%20with%20\'g\''

#year

#url = url + '&tq=select%20B%2Ccount(A)%20group%20by%20B%20order%20by%20B'

mediaList = []

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return ''

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return ''

response\_data = response.read()

response.close()

#\{"c":\[\{"v":"([^\"]\*)"\},\{"v":[^\,]\*,"f":"([^\"]\*)"\},\{"v":[^\,]\*,"f":"([^\"]\*)"\},\{"v":"([^\"]\*)"\},\{"v":"([^\"]\*)"\},\{"v":"([^\"]\*)"\},\{"v":"([^\"]\*)"\},\{"v":"([^\"]\*)"\},\{"v":"([^\"]\*)"\},\{"v":"([^\"]\*)"\},\{"v":"([^\"]\*)"\},\{"v":"([^\"]\*)"\}\]\}' ,

for r in re.finditer('\{"c"\:\[\{"v"\:"([^\"]+)"\},\{"v"\:[^\,]+\,"f"\:"([^\"]+)"\},\{"v":[^\,]+,"f"\:"([^\"]+)"\},\{"v":"([^\"]+)"\},\{"v":"([^\"]+)"\},\{"v":"([^\"]+)"\},\{"v":"([^\"]+)"\},\{"v":"([^\"]+)"\},\{"v":"([^\"]+)"\},\{"v":"([^\"]+)"\},\{"v":"([^\"]+)"\}\]\}',

response\_data, re.DOTALL):

title = r.group(1)

year = r.group(2)

rating = r.group(3)

genre = r.group(4)

plot = r.group(5)

poster = r.group(6)

fanart = r.group(7)

country = r.group(8)

set = r.group(9)

director = r.group(10)

actorList = r.group(11)

actors = []

for r in re.finditer('([^\|]+)\|' ,actorList, re.DOTALL):

actor = r.group(1)

actors.append( (actor, actor))

newPackage = package.package( file.file('', title, plot, self.service.MEDIA\_TYPE\_VIDEO, fanart,poster),folder.folder('', ''))

newPackage.file.rating = rating

newPackage.file.director = director

newPackage.file.set = set

newPackage.file.genre = genre

newPackage.file.country = country

newPackage.file.year = year

if len(actors) > 0:

newPackage.file.actors = actors

mediaList.append(newPackage)

#

return mediaList

#spreadsheet STRM

def getDirector(self, url):

for r in re.finditer('list/([^\/]+)\/' ,

url, re.DOTALL):

spreadsheetID = r.group(1)

url = 'https://docs.google.com/spreadsheets/d/'+spreadsheetID+'/gviz/tq?tqx=out.csv'

#all genre

#url = url + '&tq=select%20D%2Ccount(A)%20group%20by%20D'

url = url + '&tq=select%20J%2Ccount(A)%20group%20by%20J'

mediaList = []

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return ''

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return ''

response\_data = response.read()

response.close()

count=0;

for r in re.finditer('"c"\:\[\{"v"\:"([^\"]+)"\}' ,

response\_data, re.DOTALL):

item = r.group(1)

newPackage = package.package( None,folder.folder('CLOUD\_DB\_DIRECTOR', item))

mediaList.append(newPackage)

return mediaList

#spreadsheet STRM

def getGenre(self, url):

for r in re.finditer('list/([^\/]+)\/' ,

url, re.DOTALL):

spreadsheetID = r.group(1)

url = 'https://docs.google.com/spreadsheets/d/'+spreadsheetID+'/gviz/tq?tqx=out.csv'

#all genre

#url = url + '&tq=select%20D%2Ccount(A)%20group%20by%20D'

url = url + '&tq=select%20D%2Ccount(A)%20where%20not%20D%20contains%20\'%7C\'%20group%20by%20D'

mediaList = []

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return ''

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return ''

response\_data = response.read()

response.close()

count=0;

for r in re.finditer('"c"\:\[\{"v"\:"([^\"]+)"\}' ,

response\_data, re.DOTALL):

item = r.group(1)

newPackage = package.package( None,folder.folder('CLOUD\_DB\_GENRE', item))

mediaList.append(newPackage)

return mediaList

#spreadsheet STRM

def getStudio(self, url):

for r in re.finditer('list/([^\/]+)\/' ,

url, re.DOTALL):

spreadsheetID = r.group(1)

url = 'https://docs.google.com/spreadsheets/d/'+spreadsheetID+'/gviz/tq?tqx=out.csv'

url = url + '&tq=select%20I%2Ccount(A)%20group%20by%20I'

mediaList = []

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return ''

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return ''

response\_data = response.read()

response.close()

count=0;

for r in re.finditer('"c"\:\[\{"v"\:"([^\"]+)"\}' ,

response\_data, re.DOTALL):

item = r.group(1)

newPackage = package.package( None,folder.folder('CLOUD\_DB\_STUDIO', item))

mediaList.append(newPackage)

return mediaList

#spreadsheet STRM

# loop through alphabet

def getTitle(self, url):

mediaList = []

from string import ascii\_lowercase

for c in ascii\_lowercase:

newPackage = package.package( None,folder.folder('CLOUD\_DB\_TITLE', c))

mediaList.append(newPackage)

newPackage = package.package( None,folder.folder('CLOUD\_DB\_TITLE', '#all'))

mediaList.append(newPackage)

return mediaList

#spreadsheet STRM

def getYear(self, url):

for r in re.finditer('list/([^\/]+)\/' ,

url, re.DOTALL):

spreadsheetID = r.group(1)

url = 'https://docs.google.com/spreadsheets/d/'+spreadsheetID+'/gviz/tq?tqx=out.csv'

#all genre

#url = url + '&tq=select%20D%2Ccount(A)%20group%20by%20D'

url = url + '&tq=select%20B%2Ccount(A)%20group%20by%20B'

mediaList = []

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return ''

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return ''

response\_data = response.read()

response.close()

count=0;

for r in re.finditer('"c"\:\[\{"v"\:(\d+)' ,

response\_data, re.DOTALL):

item = r.group(1)

newPackage = package.package( None,folder.folder('CLOUD\_DB\_YEAR', item))

mediaList.append(newPackage)

return mediaList

#spreadsheet STRM

def getCountries(self, url):

for r in re.finditer('list/([^\/]+)\/' ,

url, re.DOTALL):

spreadsheetID = r.group(1)

url = 'https://docs.google.com/spreadsheets/d/'+spreadsheetID+'/gviz/tq?tqx=out.csv'

#all genre

#url = url + '&tq=select%20D%2Ccount(A)%20group%20by%20D'

url = url + '&tq=select%20H%2Ccount(A)%20group%20by%20H'

mediaList = []

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return ''

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return ''

response\_data = response.read()

response.close()

count=0;

for r in re.finditer('"c"\:\[\{"v"\:"([^\"]+)"\}' ,

response\_data, re.DOTALL):

item = r.group(1)

newPackage = package.package( None,folder.folder('CLOUD\_DB\_COUNTRY', item))

mediaList.append(newPackage)

return mediaList

def getShows(self,url,channel):

params = urllib.urlencode({'channel': channel})

url = url + '?sq=' + params

shows = {}

while True:

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

response\_data = response.read()

count=0;

for r in re.finditer('<gsx:channel>([^<]\*)</gsx:channel><gsx:month>([^<]\*)</gsx:month><gsx:day>([^<]\*)</gsx:day><gsx:weekday>([^<]\*)</gsx:weekday><gsx:hour>([^<]\*)</gsx:hour><gsx:minute>([^<]\*)</gsx:minute><gsx:show>([^<]\*)</gsx:show><gsx:order>([^<]\*)</gsx:order><gsx:includewatched>([^<]\*)</gsx:includewatched>' ,

response\_data, re.DOTALL):

shows[count] = r.groups()

#source,nfo,show,season,episode,part,watched,duration

#channel,month,day,weekday,hour,minute,show,order,includeWatched

count = count + 1

nextURL = ''

for r in re.finditer('<link rel=\'next\' type=\'[^\']+\' href=\'([^\']+)\'' ,

response\_data, re.DOTALL):

nextURL = r.groups()

response.close()

if nextURL == '':

break

else:

url = nextURL[0]

return shows

def getMedia(self,url, folderID=None, fileID=None):

if fileID is None:

params = urllib.urlencode({'folderid': folderID})

else:

params = urllib.urlencode({'fileid': fileID})

url = url + '?sq=' + params

media = {}

while True:

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

response\_data = response.read()

count=0;

# for r in re.finditer('<gsx:folderid>([^<]\*)</gsx:folderid><gsx:foldername>([^<]\*)</gsx:foldername><gsx:fileid>([^<]\*)</gsx:fileid><gsx:filename>([^<]\*)</gsx:filename><gsx:nfo>([^<]\*)</gsx:nfo><gsx:order>([^<]\*)</gsx:order><gsx:watched>([^<]\*)</gsx:watched><gsx:resume>([^<]\*)</gsx:resume>' ,

for r in re.finditer('<gsx:fileid>([^<]\*)</gsx:fileid><gsx:filename>([^<]\*)</gsx:filename><gsx:watched>([^<]\*)</gsx:watched><gsx:resume>([^<]\*)</gsx:resume>' ,

response\_data, re.DOTALL):

media[count] = r.groups()

count = count + 1

nextURL = ''

for r in re.finditer('<link rel=\'next\' type=\'[^\']+\' href=\'([^\']+)\'' ,

response\_data, re.DOTALL):

nextURL = r.groups()

response.close()

if nextURL == '':

break

else:

url = nextURL[0]

return media

def updateMediaPackage(self,url, package1=None, criteria=''):

if package1 is not None and (package1.file is None or package1.file.id is None) and package1.folder is not None and package1.folder.id is not None:

params = urllib.urlencode({'folderid': package1.folder.id})

elif package1 is not None and (package1.file is None or package1.file.id is not None) and package1.folder is not None and package1.folder.id is not None and package1.folder.id != '' :

params = str(urllib.urlencode({'folderid': package1.folder.id})) +'%20or%20'+ str(urllib.urlencode({'fileid': package1.file.id}))

elif package1 is not None and package1.file is not None and package1.file.id is not None:

params = urllib.urlencode({'fileid': package1.file.id})

elif package1 is None and criteria == 'library':

params = 'foldername!=""&orderby=column:folderid'

elif package1 is None and criteria == 'queued':

params = 'folderid=QUEUED&orderby=column:order'

elif package1 is None and criteria == 'recentwatched':

from datetime import date, timedelta

updated = str((date.today() - timedelta(1)).strftime("%Y%m%d%H%M"))

params = 'folderid!=QUEUED%20and%20watched!=""%20and%20watched>0%20and%20updated>='+updated

elif package1 is None and criteria == 'recentstarted':

from datetime import date, timedelta

updated = str((date.today() - timedelta(1)).strftime("%Y%m%d%H%M"))

params = 'folderid!=QUEUED%20and%20watched=""%20and%20resume>0%20and%20updated>='+updated

else:

return

url = url + '?sq=' + params

#url = url + '?tq=' + params

mediaList = []

while True:

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return

response\_data = response.read()

previous = ''

append = True

# for r in re.finditer('<gsx:folderid>([^<]\*)</gsx:folderid><gsx:foldername>([^<]\*)</gsx:foldername><gsx:fileid>([^<]\*)</gsx:fileid><gsx:filename>([^<]\*)</gsx:filename><gsx:nfo>([^<]\*)</gsx:nfo><gsx:order>([^<]\*)</gsx:order><gsx:watched>([^<]\*)</gsx:watched><gsx:resume>([^<]\*)</gsx:resume>' ,

for r in re.finditer('<entry>(.\*?)</entry>' ,

response\_data, re.DOTALL):

#media = r.groups()

entry = r.group()

#exp = re.compile('<gsx:([^\>]+)>(.\*)</gsx')

#exp = re.compile('<gsx:([^\>]+)>([^<]+)</')

exp = re.compile('<gsx:([^\>]+)>([^<]+)</gsx')

if package1 is None:

newPackage = package.package( file.file('', '', '', self.service.MEDIA\_TYPE\_VIDEO, '',''),folder.folder('',''))

else:

newPackage = package1

for media in exp.finditer(entry):

# not a general folder ID but another file ID

if media.group(1) == 'fileid' and newPackage.file.id != '' and newPackage.file.id != media.group(2) and media.group(2) != '':

break

elif media.group(1) == 'folderid':

newPackage.folder.id = media.group(2)

elif media.group(1) == 'foldername':

newPackage.folder.title = media.group(2)

newPackage.folder.displaytitle = media.group(2)

if criteria == 'library':

newPackage.file = None

if previous == newPackage.folder.id:

append = False

else:

append = True

previous = newPackage.folder.id

break

elif media.group(1) == 'watched':

if media.group(2) == '':

newPackage.file.playcount = 0

else:

newPackage.file.playcount = media.group(2)

elif media.group(1) == 'resume':

if media.group(2) == '':

newPackage.file.resume = 0

else:

newPackage.file.resume = media.group(2)

elif media.group(1) == 'commands':

newPackage.file.commands = media.group(2)

elif media.group(1) == 'nfo':

nfoInfo = media.group(2)

nfoInfo = re.sub('&lt;', '<', nfoInfo)

nfoInfo = re.sub('/\s?&gt;', '> </>', nfoInfo)

nfoInfo = re.sub('&gt;', '>', nfoInfo)

nfo = re.compile('<([^\>]+)>([^\<]\*)</')

for info in nfo.finditer(nfoInfo):

if info.group(1) == 'title':

newPackage.file.title = info.group(2)

elif info.group(1) == 'premiered' or info.group(1) == 'year':

newPackage.file.date = info.group(2)

elif info.group(1) == 'plot' or info.group(1) == 'description':

newPackage.file.plot = info.group(2)

elif info.group(1) == 'actors':

newPackage.file.cast = info.group(2)

elif media.group(1) == 'fanart':

newPackage.file.fanart = self.service.API\_URL +'files/' + str(media.group(2)) + '?alt=media' + '|' + self.service.getHeadersEncoded()

elif media.group(1) == 'fileid':

newPackage.file.id = media.group(2)

elif media.group(1) == 'filename':

newPackage.file.title = media.group(2)

if append:

mediaList.append(newPackage)

nextURL = ''

for r in re.finditer('<link rel=\'next\' type=\'[^\']+\' href=\'([^\']+)\'' ,

response\_data, re.DOTALL):

nextURL = r.groups()

response.close()

if nextURL == '':

break

else:

url = nextURL[0]

return mediaList

def updateMediaPackageList(self,url, folderID, mediaList):

if folderID is not None and folderID != '':

params = urllib.urlencode({'folderid': folderID})

else:

return

url = url + '?sq=' + params

#url = url + '?tq=' + params

mediaHash = {}

if mediaList:

count=0

for item in mediaList:

if item.file is not None:

mediaHash[item.file.id] = count

count = count + 1

while True:

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return

response\_data = response.read()

previous = ''

append = True

for r in re.finditer('<entry>(.\*?)</entry>' ,

response\_data, re.DOTALL):

entry = r.group()

exp = re.compile('<gsx:([^\>]+)>([^<]+)</gsx')

for media in exp.finditer(entry):

# not a general folder ID but another file ID

if media.group(1) == 'fileid' and media.group(2) not in mediaHash.keys():

break

elif media.group(1) == 'folderid' and media.group(2) not in mediaHash.keys():

break

elif media.group(1) == 'fileid':

newPackage = mediaList[mediaHash[media.group(2)]]

elif media.group(1) == 'watched':

if media.group(2) == '':

newPackage.file.playcount = 0

else:

newPackage.file.playcount = media.group(2)

elif media.group(1) == 'resume':

if media.group(2) == '':

newPackage.file.cloudResume = 0

else:

newPackage.file.cloudResume = media.group(2)

elif media.group(1) == 'commands':

newPackage.file.commands = media.group(2)

elif media.group(1) == 'nfo':

nfoInfo = media.group(2)

nfoInfo = re.sub('&lt;', '<', nfoInfo)

nfoInfo = re.sub('/\s?&gt;', '> </>', nfoInfo)

nfoInfo = re.sub('&gt;', '>', nfoInfo)

nfo = re.compile('<([^\>]+)>([^\<]\*)</')

for info in nfo.finditer(nfoInfo):

#if info.group(1) == 'title':

# newPackage.file.title = info.group(2)

if info.group(1) == 'premiered' or info.group(1) == 'year':

newPackage.file.date = info.group(2)

elif info.group(1) == 'plot' or info.group(1) == 'description':

newPackage.file.plot = info.group(2)

elif info.group(1) == 'actors':

newPackage.file.cast = info.group(2)

elif media.group(1) == 'fanart':

newPackage.file.fanart = self.service.API\_URL +'files/' + str(media.group(2)) + '?alt=media' + '|' + self.service.getHeadersEncoded()

elif media.group(1) == 'filename':

newPackage.file.title = media.group(2)

nextURL = ''

for r in re.finditer('<link rel=\'next\' type=\'[^\']+\' href=\'([^\']+)\'' ,

response\_data, re.DOTALL):

nextURL = r.groups()

response.close()

if nextURL == '':

break

else:

url = nextURL[0]

return mediaList

## not in use

def getMediaPackageList(self,url, folderName, mediaList):

if folderName is not None and folderName != '':

params = urllib.urlencode({'foldername': folderName, 'folderid': ''})

else:

return

url = url + '?sq=' + params

#url = url + '?tq=' + params

mediaHash = {}

if mediaList:

count=0

for item in mediaList:

if item.file is not None:

mediaHash[item.file.id] = count

count = count + 1

while True:

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return

response\_data = response.read()

previous = ''

append = True

for r in re.finditer('<entry>(.\*?)</entry>' ,

response\_data, re.DOTALL):

entry = r.group()

exp = re.compile('<gsx:([^\>]+)>([^<]+)</gsx')

for media in exp.finditer(entry):

# not a general folder ID but another file ID

if media.group(1) == 'fileid' and media.group(2) not in mediaHash.keys():

break

elif media.group(1) == 'fileid':

newPackage = mediaList[mediaHash[media.group(2)]]

elif media.group(1) == 'watched':

if media.group(2) == '':

newPackage.file.playcount = 0

else:

newPackage.file.playcount = media.group(2)

elif media.group(1) == 'resume':

if media.group(2) == '':

newPackage.file.cloudResume = 0

else:

newPackage.file.cloudResume = media.group(2)

elif media.group(1) == 'commands':

newPackage.file.commands = media.group(2)

elif media.group(1) == 'nfo':

nfoInfo = media.group(2)

nfoInfo = re.sub('&lt;', '<', nfoInfo)

nfoInfo = re.sub('/\s?&gt;', '> </>', nfoInfo)

nfoInfo = re.sub('&gt;', '>', nfoInfo)

nfo = re.compile('<([^\>]+)>([^\<]\*)</')

for info in nfo.finditer(nfoInfo):

if info.group(1) == 'title':

newPackage.file.title = info.group(2)

elif info.group(1) == 'premiered' or info.group(1) == 'year':

newPackage.file.date = info.group(2)

elif info.group(1) == 'plot' or info.group(1) == 'description':

newPackage.file.plot = info.group(2)

elif info.group(1) == 'actors':

newPackage.file.cast = info.group(2)

elif media.group(1) == 'fanart':

newPackage.file.fanart = self.service.API\_URL +'files/' + str(media.group(2)) + '?alt=media' + '|' + self.service.getHeadersEncoded()

elif media.group(1) == 'filename':

newPackage.file.title = media.group(2)

nextURL = ''

for r in re.finditer('<link rel=\'next\' type=\'[^\']+\' href=\'([^\']+)\'' ,

response\_data, re.DOTALL):

nextURL = r.groups()

response.close()

if nextURL == '':

break

else:

url = nextURL[0]

return mediaList

def getMediaInformation(self,url,folderID):

params = urllib.urlencode({'folderuid': folderID})

url = url + '?sq=' + params

media = {}

while True:

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

return

response\_data = response.read()

count=0;

for r in re.finditer('<gsx:foldername>([^<]\*)</gsx:foldername><gsx:folderuid>([^<]\*)</gsx:folderuid><gsx:filename>([^<]\*)</gsx:filename><gsx:fileuid>([^<]\*)</gsx:fileuid><gsx:season>([^<]\*)</gsx:season><gsx:episode>([^<]\*)</gsx:episode><gsx:watched>([^<]\*)</gsx:watched><gsx:sequence>([^<]\*)</gsx:sequence>' ,

response\_data, re.DOTALL):

media[count] = r.groups()

count = count + 1

nextURL = ''

for r in re.finditer('<link rel=\'next\' type=\'[^\']+\' href=\'([^\']+)\'' ,

response\_data, re.DOTALL):

nextURL = r.groups()

response.close()

if nextURL == '':

break

else:

url = nextURL[0]

return media

def getVideo(self,url,show):

params = urllib.urlencode({'show': show})

url = url + '?sq=' + params + '+and+watched=0'

shows = {}

while True:

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

response\_data = response.read()

count=0;

for r in re.finditer('<entry[^\>]\*>.\*?<gsx:source>([^<]\*)</gsx:source><gsx:nfo>([^<]\*)</gsx:nfo><gsx:show>([^<]\*)</gsx:show><gsx:season>([^<]\*)</gsx:season><gsx:episode>([^<]\*)</gsx:episode><gsx:part>([^<]\*)</gsx:part><gsx:watched>([^<]\*)</gsx:watched><gsx:duration>([^<]\*)</gsx:duration></entry>' ,

response\_data, re.DOTALL):

shows[count] = r.groups()

#source,nfo,show,season,episode,part,watched,duration

count = count + 1

nextURL = ''

for r in re.finditer('<link rel=\'next\' type=\'[^\']+\' href=\'([^\']+)\'' ,

response\_data, re.DOTALL):

nextURL = r.groups()

response.close()

if nextURL == '':

break

else:

url = nextURL[0]

return shows

def setVideoWatched(self,url,source):

# import urllib

# from cookielib import CookieJar

# cj = CookieJar()

# opener = urllib2.build\_opener(urllib2.HTTPCookieProcessor(cj))

# urllib2.install\_opener(opener)

source = re.sub(' ', '+', source)

# params = urllib.urlencode(source)

url = url + '?sq=source="' + source +'"'

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

# response = opener.open(url, None,urllib.urlencode(header))

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

response\_data = response.read()

response.close()

editURL=''

for r in re.finditer('<link rel=\'(edit)\' type=\'application/atom\+xml\' href=\'([^\']+)\'/>' ,

response\_data, re.DOTALL):

(x,editURL) = r.groups(1)

for r in re.finditer('<link rel=\'edit\' [^\>]+>(.\*?</entry>)' ,

response\_data, re.DOTALL):

entry = r.group(1)

entry = re.sub('<gsx:watched>([^\<]\*)</gsx:watched>', '<gsx:watched>1</gsx:watched>', entry)

# entry = re.sub(' gd\:etag[^\>]+>', ' xmlns="http://www.w3.org/2005/Atom" xmlns:gs="http://schemas.google.com/spreadsheets/2006" xmlns:gsx="http://schemas.google.com/spreadsheets/2006/extended">', entry)

# entry = re.sub('<entry>', '<?xml version=\'1.0\' encoding=\'UTF-8\'?><entry xmlns="http://www.w3.org/2005/Atom" xmlns:gsx="http://schemas.google.com/spreadsheets/2006/extended">', entry)

#entry = re.sub('<entry>', '<?xml version=\'1.0\' encoding=\'UTF-8\'?><entry xmlns="http://www.w3.org/2005/Atom" xmlns:gsx="http://schemas.google.com/spreadsheets/2006/extended"> ', entry)

entry = '<?xml version=\'1.0\' encoding=\'UTF-8\'?><entry xmlns="http://www.w3.org/2005/Atom" xmlns:gs="http://schemas.google.com/spreadsheets/2006" xmlns:gsx="http://schemas.google.com/spreadsheets/2006/extended">' + entry

# entry = "<?xml version='1.0' encoding='UTF-8'?><entry xmlns='http://www.w3.org/2005/Atom' xmlns:gsx='http://schemas.google.com/spreadsheets/2006/extended'><id>https://spreadsheets.google.com/feeds/list/147ajW3jRGUTwcuBSLx5dYw5ar17fo9NPtu8azHa3j0w/od6/private/full/1lcxsw</id><updated>2015-05-01T18:49:50.299Z</updated><category scheme='http://schemas.google.com/spreadsheets/2006' term='http://schemas.google.com/spreadsheets/2006#list'/><title type='text'>S3E12 - The Red Dot.avi-0002</title><content type='text'>nfo: test.nfo, show: Seinfeld, season: 3, episode: 1, part: 1, watched: 0, duration: 1</content><link rel='self' type='application/atom+xml' href='https://spreadsheets.google.com/feeds/list/147ajW3jRGUTwcuBSLx5dYw5ar17fo9NPtu8azHa3j0w/od6/private/full/1lcxsw'/><link rel='edit' type='application/atom+xml' href='https://spreadsheets.google.com/feeds/list/147ajW3jRGUTwcuBSLx5dYw5ar17fo9NPtu8azHa3j0w/od6/private/full/1lcxsw/in881g9gmnffm'/><gsx:source>S3E12 - The Red Dot.avi-0002</gsx:source><gsx:nfo>test.nfo</gsx:nfo><gsx:show>Seinfeld</gsx:show><gsx:season>3</gsx:season><gsx:episode>1</gsx:episode><gsx:part>1</gsx:part><gsx:watched>0</gsx:watched><gsx:duration>1</gsx:duration></entry>"

#xmlns:gsx='http://schemas.google.com/spreadsheets/2006/extended'

# entry = " <?xml version='1.0' encoding='UTF-8'?><feed xmlns='http://www.w3.org/2005/Atom' xmlns:openSearch='http://a9.com/-/spec/opensearchrss/1.0/' xmlns:gsx='http://schemas.google.com/spreadsheets/2006/extended' xmlns:gd=\"http://schemas.google.com/g/2005\">"+entry

# entry = '<feed xmlns="http://www.w3.org/2005/Atom" xmlns:openSearch="http://a9.com/-/spec/opensearch/1.1/" xmlns:gsx="http://schemas.google.com/spreadsheets/2006/extended" xmlns:gd="http://schemas.google.com/g/2005" gd:etag=\'W/"D0cERnk-eip7ImA9WBBXGEg."\'><entry> <id> https://spreadsheets.google.com/feeds/worksheets/key/private/full/worksheetId </id> <updated>2007-07-30T18:51:30.666Z</updated> <category scheme="http://schemas.google.com/spreadsheets/2006" term="http://schemas.google.com/spreadsheets/2006#worksheet"/> <title type="text">Income</title> <content type="text">Expenses</content> <link rel="http://schemas.google.com/spreadsheets/2006#listfeed" type="application/atom+xml" href="https://spreadsheets.google.com/feeds/list/key/worksheetId/private/full"/> <link rel="http://schemas.google.com/spreadsheets/2006#cellsfeed" type="application/atom+xml" href="https://spreadsheets.google.com/feeds/cells/key/worksheetId/private/full"/> <link rel="self" type="application/atom+xml" href="https://spreadsheets.google.com/feeds/worksheets/key/private/full/worksheetId"/> <link rel="edit" type="application/atom+xml" href="https://spreadsheets.google.com/feeds/worksheets/key/private/full/worksheetId/version"/> <gs:rowCount>45</gs:rowCount> <gs:colCount>15</gs:colCount></entry>'

# req = urllib2.Request(editURL, entry, header)

# urllib2.HTTPHandler(debuglevel=1)

# req.get\_method = lambda: 'PUT'

req = urllib2.Request(editURL, entry, self.service.getHeadersList(isPOST=True))

req.get\_method = lambda: 'PUT'

# req.get\_method = lambda: 'DELETE'

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e.read()), xbmc.LOGERROR)

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e.read()), xbmc.LOGERROR)

response\_data = response.read()

response.close()

def setMediaStatus(self, url, package, resume='', watched=''):

import time

updated = time.strftime("%Y%m%d%H%M")

newurl = url + '?sq=fileid="' + str(package.file.id) +'"'

req = urllib2.Request(newurl, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

# response = opener.open(url, None,urllib.urlencode(header))

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

response\_data = response.read()

response.close()

editURL=''

for r in re.finditer('<link rel=\'edit\' type=\'application/atom\+xml\' href=\'([^\']+)\'/>' ,

response\_data, re.DOTALL):

editURL = r.group(1)

for r in re.finditer('<link rel=\'edit\' [^\>]+>(.\*?</entry>)' ,

response\_data, re.DOTALL):

entry = r.group(1)

if editURL != '':

if resume != '':

entry = re.sub('<gsx:resume>([^\<]\*)</gsx:resume>', '<gsx:resume>'+str(resume)+'</gsx:resume>', entry)

if watched != '':

entry = re.sub('<gsx:watched>([^\<]\*)</gsx:watched>', '<gsx:watched>'+str(watched)+'</gsx:watched>', entry)

entry = re.sub('<gsx:updated>([^\<]\*)</gsx:updated>', '<gsx:updated>'+str(updated)+'</gsx:updated>', entry)

entry = '<?xml version=\'1.0\' encoding=\'UTF-8\'?><entry xmlns="http://www.w3.org/2005/Atom" xmlns:gs="http://schemas.google.com/spreadsheets/2006" xmlns:gsx="http://schemas.google.com/spreadsheets/2006/extended">' + entry

req = urllib2.Request(editURL, entry, self.service.getHeadersList(isPOST=True))

req.get\_method = lambda: 'PUT'

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e.read()), xbmc.LOGERROR)

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e.read()), xbmc.LOGERROR)

response\_data = response.read()

response.close()

else:

if resume != '' and watched != '':

self.createMediaStatus(url,package,resume,watched, updated=updated)

elif resume != '' and watched == '':

self.createMediaStatus(url,package,resume=resume, updated=updated)

elif resume == '' and watched != '':

self.createMediaStatus(url,package,watched=watched, updated=updated)

else:

self.createMediaStatus(url,package, updated=updated)

def getChannels(self,url):

params = urllib.urlencode({'orderby': 'channel'})

url = url + '?' + params

channels = []

count=0

while True:

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

if e.code == 403 or e.code == 401:

self.service.refreshToken()

req = urllib2.Request(url, None, self.service.getHeadersList())

try:

response = urllib2.urlopen(req)

except urllib2.URLError, e:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

else:

xbmc.log(self.addon.getAddonInfo('name') + ': ' + str(e), xbmc.LOGERROR)

response\_data = response.read()

for r in re.finditer('<gsx:channel>([^<]\*)</gsx:channel>' ,

response\_data, re.DOTALL):

(channel) = r.groups()

#channel,month,day,weekday,hour,minute,show,order,includeWatched

if not channels.\_\_contains\_\_(channel[0]):

channels.append(channel[0])

count = count + 1

nextURL = ''

for r in re.finditer('<link rel=\'next\' type=\'[^\']+\' href=\'([^\']+)\'' ,

response\_data, re.DOTALL):

nextURL = r.groups()

response.close()

if nextURL == '':

break

else:

url = nextURL[0]

return channels