'''

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/>.

'''

# cloudservice - required python modules

import sys

import cgi

import os

import re

# cloudservice - standard XBMC modules

import xbmc, xbmcgui, xbmcplugin, xbmcaddon, xbmcvfs

from resources.lib import settings

from resources.lib import offlinefile

# global variables

import addon\_parameters

addon = addon\_parameters.addon

PLUGIN\_URL = sys.argv[0]

plugin\_handle = None

try:

plugin\_handle = int(sys.argv[1])

except:pass

def decode(data):

return re.sub("&#(\d+)(;|(?=\s))", \_callback, data).strip()

def decode\_dict(data):

for k, v in data.items():

if type(v) is str or type(v) is unicode:

data[k] = decode(v)

return data

#http://stackoverflow.com/questions/1208916/decoding-html-entities-with-python/1208931#1208931

def \_callback(matches):

id = matches.group(1)

try:

return unichr(int(id))

except:

return id

##

# load eclipse debugger

# parameters: none

##

def debugger():

try:

remote\_debugger = settings.getSetting('remote\_debugger')

remote\_debugger\_host = settings.getSetting('remote\_debugger\_host')

# append pydev remote debugger

if remote\_debugger == 'true':

# Make pydev debugger works for auto reload.

# Note pydevd module need to be copied in XBMC\system\python\Lib\pysrc

import pysrc.pydevd as pydevd

# stdoutToServer and stderrToServer redirect stdout and stderr to eclipse console

pydevd.settrace(remote\_debugger\_host, stdoutToServer=True, stderrToServer=True)

except ImportError:

xbmc.log(addon.getLocalizedString(30016), xbmc.LOGERROR)

sys.exit(1)

except :

pass

##

# add a menu to a directory screen

# parameters: url to resolve, title to display, optional: icon, fanart, total\_items, instance name

##

def addMenu(url, title, img='', fanart='', total\_items=0, instanceName=''):

# listitem = xbmcgui.ListItem(decode(title), iconImage=img, thumbnailImage=img)

listitem = xbmcgui.ListItem(title, iconImage=img, thumbnailImage=img)

if not fanart:

fanart = addon.getAddonInfo('path') + '/fanart.jpg'

listitem.setProperty('fanart\_image', fanart)

# disallow play controls on menus

listitem.setProperty('IsPlayable', 'false')

if instanceName != '':

cm=[]

cm.append(( addon.getLocalizedString(30159), 'XBMC.RunPlugin('+PLUGIN\_URL+ '?mode=delete&instance='+instanceName+')' ))

listitem.addContextMenuItems(cm, True)

xbmcplugin.addDirectoryItem(plugin\_handle, url, listitem,

isFolder=True, totalItems=total\_items)

##

# Providing a context type, return what content to display based on user's preferences

# parameters: current context type plugin was invoked in (audio, video, photos)

##

def getContentType(contextType,encfs):

#contentType

#video context

# 0 video

# 1 video and music

# 2 everything

# 9 encrypted video

#

#music context

# 3 music

# 4 everything

# 10 encrypted video

#

#photo context

# 5 photo

# 6 music and photos

# 7 everything

# 11 encrypted photo

contentType = 0

if contextType == 'video':

if encfs:

contentTypeDecider = int(settings.getSetting('context\_evideo',0))

if contentTypeDecider == 1:

contentType = 8

else:

contentType = 9

else:

contentTypeDecider = int(settings.getSetting('context\_video',0))

if contentTypeDecider == 2:

contentType = 2

elif contentTypeDecider == 1:

contentType = 1

else:

contentType = 0

# cloudservice - sorting options

xbmcplugin.addSortMethod(int(sys.argv[1]), xbmcplugin.SORT\_METHOD\_EPISODE)

xbmcplugin.setContent(int(sys.argv[1]), 'movies')

elif contextType == 'audio':

if encfs:

contentTypeDecider = int(settings.getSetting('context\_emusic',0))

if contentTypeDecider == 1:

contentType = 8

else:

contentType = 10

else:

contentTypeDecider = int(settings.getSetting('context\_music', 0))

if contentTypeDecider == 1:

contentType = 4

else:

contentType = 3

# cloudservice - sorting options

xbmcplugin.addSortMethod(int(sys.argv[1]), xbmcplugin.SORT\_METHOD\_TRACKNUM)

elif contextType == 'image':

if encfs:

contentTypeDecider = int(settings.getSetting('context\_ephoto',0))

if contentTypeDecider == 1:

contentType = 8

else:

contentType = 11

else:

contentTypeDecider = int(settings.getSetting('context\_photo', 0))

if contentTypeDecider == 2:

contentType = 7

elif contentTypeDecider == 1:

contentType = 6

else:

contentType = 5

# show all (for encfs)

elif contextType == 'all':

contentType = 8

return contentType

##

# get a list of offline files

##

def getOfflineFileList(cachePath):

localFiles = []

#workaround for this issue: https://github.com/xbmc/xbmc/pull/8531

if xbmcvfs.exists(cachePath) or os.path.exists(cachePath):

dirs,files = xbmcvfs.listdir(cachePath)

for dir in dirs:

subdir,subfiles = xbmcvfs.listdir(str(cachePath) + '/' + str(dir))

for file in subfiles:

if bool(re.search('\.stream\.mp4', file)):

try:

nameFile = xbmcvfs.File(str(cachePath) + '/' + str(dir) + '/' + str(dir) + '.name')

filename = nameFile.read()

nameFile.close()

except:

filename = file

try:

nameFile = xbmcvfs.File(str(cachePath) + '/' + str(dir) + '/' + str(os.path.splitext(file)[0]) + '.resolution')

resolution = nameFile.read()

nameFile.close()

except:

resolution = file

offlineFile = offlinefile.offlinefile(filename, str(cachePath) + '/' + str(dir) +'.jpg', resolution.rstrip(), str(cachePath) + '/' + str(dir) + '/' + str(os.path.splitext(file)[0]) + '.mp4')

localFiles.append(offlineFile)

return localFiles

##

# Add a media file to a directory listing screen

# parameters: package, context type, whether file is encfs, encfs:decryption path, encfs:encryption path

##

def addOfflineMediaFile(offlinefile):

listitem = xbmcgui.ListItem(offlinefile.title, iconImage=offlinefile.thumbnail,

thumbnailImage=offlinefile.thumbnail)

if offlinefile.resolution == 'original':

infolabels = decode\_dict({ 'title' : offlinefile.title})

else:

infolabels = decode\_dict({ 'title' : offlinefile.title + ' - ' + offlinefile.resolution })

listitem.setInfo('Video', infolabels)

listitem.setProperty('IsPlayable', 'true')

xbmcplugin.addDirectoryItem(plugin\_handle, offlinefile.playbackpath, listitem,

isFolder=False, totalItems=0)

return offlinefile.playbackpath

##

# Calculate the number of accounts defined in settings

# parameters: the account type (usually plugin name)

##

def numberOfAccounts(accountType):

return 9

count = 1

max\_count = int(settings.getSetting(accountType+'\_numaccounts',9))

actualCount = 0

while True:

try:

if settings.getSetting(accountType+str(count)+'\_username') != '':

actualCount = actualCount + 1

except:

break

if count == max\_count:

break

count = count + 1

return actualCount

##

# Delete an account, enroll an account or refresh the current listings

# parameters: mode

##

def accountActions(addon, PLUGIN\_NAME, mode, instanceName, numberOfAccounts):

if mode == 'dummy':

xbmc.executebuiltin("XBMC.Container.Refresh")

# delete the configuration for the specified account

elif mode == 'delete':

#\*\*\* old - needs to be re-written

if instanceName != '':

try:

# gdrive specific \*\*\*

addon.setSetting(instanceName + '\_username', '')

addon.setSetting(instanceName + '\_code', '')

addon.setSetting(instanceName + '\_client\_id', '')

addon.setSetting(instanceName + '\_client\_secret', '')

addon.setSetting(instanceName + '\_url', '')

addon.setSetting(instanceName + '\_password', '')

addon.setSetting(instanceName + '\_passcode', '')

addon.setSetting(instanceName + '\_auth\_access\_token', '')

addon.setSetting(instanceName + '\_auth\_refresh\_token', '')

addon.setSetting(instanceName + '\_spreadsheetname', '')

addon.setSetting(instanceName + '\_spreadsheetname', '')

addon.setSetting(instanceName + '\_spreadsheet', '')

# \*\*\*

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30158))

except:

#error: instance doesn't exist

pass

xbmc.executebuiltin("XBMC.Container.Refresh")

# enroll a new account

elif mode == 'enroll':

invokedUsername = settings.getParameter('username')

code = settings.getParameter('code', '')

if code == '':

options = []

options.append('Google Apps')

ret = xbmcgui.Dialog().select('select type', options)

invokedUsername = ''

password = ''

if ret == 0:

try:

dialog = xbmcgui.Dialog()

invokedUsername = dialog.input('username', type=xbmcgui.INPUT\_ALPHANUM)

passcode = dialog.input('passcode', type=xbmcgui.INPUT\_ALPHANUM)

except:

pass

count = 1

loop = True

while loop:

instanceName = PLUGIN\_NAME+str(count)

try:

username = settings.getSetting(instanceName+'\_username')

if username == invokedUsername:

addon.setSetting(instanceName + '\_type', str(4))

addon.setSetting(instanceName + '\_username', str(invokedUsername))

addon.setSetting(instanceName + '\_passcode', str(passcode))

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30118), invokedUsername)

loop = False

elif username == '':

addon.setSetting(instanceName + '\_type', str(4))

addon.setSetting(instanceName + '\_username', str(invokedUsername))

addon.setSetting(instanceName + '\_passcode', str(passcode))

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30118), invokedUsername)

loop = False

except:

pass

if count == numberOfAccounts:

#fallback on first defined account

addon.setSetting(instanceName + '\_type', str(4))

addon.setSetting(instanceName + '\_username', invokedUsername)

addon.setSetting(instanceName + '\_passcode', str(passcode))

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30118), invokedUsername)

loop = False

count = count + 1

else:

count = 1

loop = True

while loop:

instanceName = PLUGIN\_NAME+str(count)

try:

username = settings.getSetting(instanceName+'\_username')

if username == invokedUsername:

addon.setSetting(instanceName + '\_type', str(1))

addon.setSetting(instanceName + '\_code', str(code))

addon.setSetting(instanceName + '\_username', str(invokedUsername))

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30118), invokedUsername)

loop = False

elif username == '':

addon.setSetting(instanceName + '\_type', str(1))

addon.setSetting(instanceName + '\_code', str(code))

addon.setSetting(instanceName + '\_username', str(invokedUsername))

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30118), invokedUsername)

loop = False

except:

pass

if count == numberOfAccounts:

#fallback on first defined account

addon.setSetting(instanceName + '\_type', str(1))

addon.setSetting(instanceName + '\_code', code)

addon.setSetting(instanceName + '\_username', invokedUsername)

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30118), invokedUsername)

loop = False

count = count + 1

##

# Delete an account, enroll an account or refresh the current listings

# parameters: addon, plugin name, mode, instance name, user provided username, number of accounts, current context

# returns: selected instance name

##

def getInstanceName(addon, PLUGIN\_NAME, mode, instanceName, invokedUsername, numberOfAccounts, contextType):

# show list of services

if mode == 'delete' or mode == 'dummy':

count = 1

elif numberOfAccounts > 1 and instanceName == '' and invokedUsername == '' and mode == 'main':

addMenu(PLUGIN\_URL+'?mode=enroll&content\_type='+str(contextType),'[enroll account]')

if contextType != 'image':

path = settings.getSetting('cache\_folder')

if path != '' and (xbmcvfs.exists(path) or os.path.exists(path)):

addMenu(PLUGIN\_URL+'?mode=offline&content\_type='+str(contextType),'<offline media>')

if contextType == 'image':

path = settings.getSetting('photo\_folder')

if path != '' and (xbmcvfs.exists(path) or os.path.exists(path)):

addMenu(path,'<offline photos>')

path = settings.getSetting('encfs\_target')

if path != '' and (xbmcvfs.exists(path) or os.path.exists(path)):

addMenu(path,'<offline encfs>')

mode = ''

count = 1

while True:

instanceName = PLUGIN\_NAME+str(count)

try:

username = settings.getSetting(instanceName+'\_username')

if username != '':

addMenu(PLUGIN\_URL+'?mode=main&content\_type='+str(contextType)+'&instance='+str(instanceName),username, instanceName=instanceName)

except:

pass

if count == numberOfAccounts:

break

count = count + 1

return None

# spreadshetModule = getSetting('library', False)

# libraryAccount = getSetting('library\_account')

# if spreadshetModule:

# addMenu(PLUGIN\_URL+'?mode=kiosk&content\_type='+str(contextType)+'&instance='+PLUGIN\_NAME+str(libraryAccount),'[kiosk mode]')

elif instanceName == '' and invokedUsername == '' and numberOfAccounts == 1:

count = 1

options = []

accounts = []

for count in range (1, numberOfAccounts+1):

instanceName = PLUGIN\_NAME+str(count)

try:

username = settings.getSetting(instanceName+'\_username')

if username != '':

options.append(username)

accounts.append(instanceName)

if username != '':

return instanceName

except:

return instanceName

#fallback on first defined account

return accounts[0]

# no accounts defined and url provided; assume public

elif numberOfAccounts == 0 and mode=='streamurl':

return None

# offline mode

elif mode=='offline':

return None

# no accounts defined

elif numberOfAccounts == 0:

#legacy account conversion

try:

username = settings.getSetting('username')

if username != '':

addon.setSetting(PLUGIN\_NAME+'1\_username', username)

addon.setSetting(PLUGIN\_NAME+'1\_password', settings,getSetting('password'))

addon.setSetting(PLUGIN\_NAME+'1\_auth\_writely', settings.getSetting('auth\_writely'))

addon.setSetting(PLUGIN\_NAME+'1\_auth\_wise', settings.getSetting('auth\_wise'))

addon.setSetting('username', '')

addon.setSetting('password', '')

addon.setSetting('auth\_writely', '')

addon.setSetting('auth\_wise', '')

else:

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30015))

xbmcplugin.endOfDirectory(plugin\_handle)

except :

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30015))

xbmcplugin.endOfDirectory(plugin\_handle)

return instanceName

# show entries of a single account (such as folder)

elif instanceName != '':

return instanceName

elif invokedUsername != '':

options = []

accounts = []

for count in range (1, numberOfAccounts+1):

instanceName = PLUGIN\_NAME+str(count)

try:

username = settings.getSetting(instanceName+'\_username')

if username != '':

options.append(username)

accounts.append(instanceName)

if username == invokedUsername:

return instanceName

except:

return instanceName

#fallback on first defined account

return accounts[0]

#prompt before playback

else:

options = []

accounts = []

# url provided; provide public option

if mode=='streamurl':

options.append('\*public')

accounts.append('public')

for count in range (1, numberOfAccounts+1):

instanceName = PLUGIN\_NAME+str(count)

try:

username = settings.getSetting(instanceName+'\_username',10)

if username != '':

options.append(username)

accounts.append(instanceName)

except:

break

# url provided; provide public option

if mode=='streamurl':

options.append('public')

accounts.append('public')

ret = xbmcgui.Dialog().select(addon.getLocalizedString(30120), options)

#fallback on first defined account

if accounts[ret] == 'public':

return None

else:

return accounts[ret]