################################################################################

# Copyright (C) 2015 Surfacingx #

# #

# 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 2, 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 XBMC; see the file COPYING. If not, write to #

# the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA. #

# http://www.gnu.org/copyleft/gpl.html #

################################################################################

import xbmc, xbmcaddon, xbmcgui, xbmcplugin, os, sys, xbmcvfs, glob

import shutil

import urllib2,urllib

import re

import uservar

import time

try: from sqlite3 import dbapi2 as database

except: from pysqlite2 import dbapi2 as database

from datetime import date, datetime, timedelta

from resources.libs import wizard as wiz

ADDON\_ID = uservar.ADDON\_ID

ADDONTITLE = uservar.ADDONTITLE

ADDON = wiz.addonId(ADDON\_ID)

DIALOG = xbmcgui.Dialog()

HOME = xbmc.translatePath('special://home/')

ADDONS = os.path.join(HOME, 'addons')

USERDATA = os.path.join(HOME, 'userdata')

PLUGIN = os.path.join(ADDONS, ADDON\_ID)

PACKAGES = os.path.join(ADDONS, 'packages')

ADDONDATA = os.path.join(USERDATA, 'addon\_data', ADDON\_ID)

ADDOND = os.path.join(USERDATA, 'addon\_data')

REALFOLD = os.path.join(ADDONDATA, 'Debrid')

ICON = os.path.join(PLUGIN, 'icon.png')

TODAY = date.today()

TOMORROW = TODAY + timedelta(days=1)

THREEDAYS = TODAY + timedelta(days=3)

KEEPTRAKT = wiz.getS('keepdebrid')

REALSAVE = wiz.getS('debridlastsave')

COLOR1 = uservar.COLOR1

COLOR2 = uservar.COLOR2

ORDER = ['specto', 'mrknow', 'url']

DEBRIDID = {

'specto': {

'name' : 'Specto',

'plugin' : 'plugin.video.specto',

'saved' : 'realspecto',

'path' : os.path.join(ADDONS, 'plugin.video.specto'),

'icon' : os.path.join(ADDONS, 'plugin.video.specto', 'icon.png'),

'fanart' : os.path.join(ADDONS, 'plugin.video.specto', 'fanart.jpg'),

'file' : os.path.join(REALFOLD, 'specto\_debrid'),

'settings' : os.path.join(ADDOND, 'plugin.video.specto', 'settings.xml'),

'default' : 'realdebrid\_client\_id',

'data' : ['realdebrid\_auth', 'realdebrid\_token', 'realdebrid\_refresh', 'realdebrid\_client\_id', 'realdebrid\_client\_secret'],

'activate' : 'RunPlugin(plugin://plugin.video.specto/?action=realdebridauth)'},

'mrknow': {

'name' : 'Mr Know URL Resolver',

'plugin' : 'script.mrknow.urlresolver',

'saved' : 'mrknowresolver',

'path' : os.path.join(ADDONS, 'script.mrknow.urlresolver'),

'icon' : os.path.join(ADDONS, 'script.mrknow.urlresolver', 'icon.png'),

'fanart' : os.path.join(ADDONS, 'script.mrknow.urlresolver', 'fanart.jpg'),

'file' : os.path.join(REALFOLD, 'mrknow\_debrid'),

'settings' : os.path.join(ADDOND, 'script.mrknow.urlresolver', 'settings.xml'),

'default' : 'RealDebridResolver\_client\_id',

'data' : ['RealDebridResolver\_enabled', 'RealDebridResolver\_priority', 'RealDebridResolver\_autopick', 'RealDebridResolver\_token', 'RealDebridResolver\_refresh', 'RealDebridResolver\_client\_id', 'RealDebridResolver\_client\_secret'],

'activate' : 'RunPlugin(plugin://script.module.urlresolver/?mode=auth\_rd)'},

'url': {

'name' : 'URL Resolver',

'plugin' : 'script.module.urlresolver',

'saved' : 'urlresolver',

'path' : os.path.join(ADDONS, 'script.module.urlresolver'),

'icon' : os.path.join(ADDONS, 'script.module.urlresolver', 'icon.png'),

'fanart' : os.path.join(ADDONS, 'script.module.urlresolver', 'fanart.jpg'),

'file' : os.path.join(REALFOLD, 'url\_debrid'),

'settings' : os.path.join(ADDOND, 'script.module.urlresolver', 'settings.xml'),

'default' : 'RealDebridResolver\_client\_id',

'data' : ['RealDebridResolver\_enabled', 'RealDebridResolver\_priority', 'RealDebridResolver\_autopick', 'RealDebridResolver\_token', 'RealDebridResolver\_refresh', 'RealDebridResolver\_client\_id', 'RealDebridResolver\_client\_secret'],

'activate' : 'RunPlugin(plugin://script.module.urlresolver/?mode=auth\_rd)'}

}

def debridUser(who):

user=None

if DEBRIDID[who]:

if os.path.exists(DEBRIDID[who]['path']):

try:

add = wiz.addonId(DEBRIDID[who]['plugin'])

user = add.getSetting(DEBRIDID[who]['default'])

except:

pass

return user

def debridIt(do, who):

if not os.path.exists(ADDONDATA): os.makedirs(ADDONDATA)

if not os.path.exists(REALFOLD): os.makedirs(REALFOLD)

if who == 'all':

for log in ORDER:

if os.path.exists(DEBRIDID[log]['path']):

try:

addonid = wiz.addonId(DEBRIDID[log]['plugin'])

default = DEBRIDID[log]['default']

user = addonid.getSetting(default)

if user == '' and do == 'update': continue

updateDebrid(do, log)

except: pass

else: wiz.log('[Real Debrid Data] %s(%s) is not installed' % (DEBRIDID[log]['name'],DEBRIDID[log]['plugin']), xbmc.LOGERROR)

wiz.setS('debridlastsave', str(THREEDAYS))

else:

if DEBRIDID[who]:

if os.path.exists(DEBRIDID[who]['path']):

updateDebrid(do, who)

else: wiz.log('[Real Debrid Data] Invalid Entry: %s' % who, xbmc.LOGERROR)

def clearSaved(who, over=False):

if who == 'all':

for debrid in DEBRIDID:

clearSaved(debrid, True)

elif DEBRIDID[who]:

file = DEBRIDID[who]['file']

if os.path.exists(file):

os.remove(file)

wiz.LogNotify('[COLOR %s]%s[/COLOR]' % (COLOR1, DEBRIDID[who]['name']),'[COLOR %s]Real Debrid Data: Removed![/COLOR]' % COLOR2, 2000, DEBRIDID[who]['icon'])

wiz.setS(DEBRIDID[who]['saved'], '')

if over == False: wiz.refresh()

def updateDebrid(do, who):

file = DEBRIDID[who]['file']

settings = DEBRIDID[who]['settings']

data = DEBRIDID[who]['data']

addonid = wiz.addonId(DEBRIDID[who]['plugin'])

saved = DEBRIDID[who]['saved']

default = DEBRIDID[who]['default']

user = addonid.getSetting(default)

suser = wiz.getS(saved)

name = DEBRIDID[who]['name']

icon = DEBRIDID[who]['icon']

if do == 'update':

if not user == '':

try:

with open(file, 'w') as f:

for debrid in data:

f.write('<debrid>\n\t<id>%s</id>\n\t<value>%s</value>\n</debrid>\n' % (debrid, addonid.getSetting(debrid)))

f.close()

user = addonid.getSetting(default)

wiz.setS(saved, user)

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, name), '[COLOR %s]Real Debrid Data: Saved![/COLOR]' % COLOR2, 2000, icon)

except Exception, e:

wiz.log("[Real Debrid Data] Unable to Update %s (%s)" % (who, str(e)), xbmc.LOGERROR)

else: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, name), '[COLOR %s]Real Debrid Data: Not Registered![/COLOR]' % COLOR2, 2000, icon)

elif do == 'restore':

if os.path.exists(file):

f = open(file,mode='r'); g = f.read().replace('\n','').replace('\r','').replace('\t',''); f.close();

match = re.compile('<debrid><id>(.+?)</id><value>(.+?)</value></debrid>').findall(g)

try:

if len(match) > 0:

for debrid, value in match:

addonid.setSetting(debrid, value)

user = addonid.getSetting(default)

wiz.setS(saved, user)

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, name), '[COLOR %s]Real Debrid: Restored![/COLOR]' % COLOR2, 2000, icon)

except Exception, e:

wiz.log("[Real Debrid Data] Unable to Restore %s (%s)" % (who, str(e)), xbmc.LOGERROR)

#else: wiz.LogNotify(name,'Real Debrid Data: [COLOR red]Not Found![/COLOR]', 2000, icon)

elif do == 'clearaddon':

wiz.log('%s SETTINGS: %s' % (name, settings), xbmc.LOGDEBUG)

if os.path.exists(settings):

try:

f = open(settings, "r"); lines = f.readlines(); f.close()

f = open(settings, "w")

for line in lines:

match = wiz.parseDOM(line, 'setting', ret='id')

if len(match) == 0: f.write(line)

else:

if match[0] not in data: f.write(line)

else: wiz.log('Removing Line: %s' % line, xbmc.LOGNOTICE)

f.close()

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, name),'[COLOR %s]Addon Data: Cleared![/COLOR]' % COLOR2, 2000, icon)

except Exception, e:

wiz.log("[Trakt Data] Unable to Clear Addon %s (%s)" % (who, str(e)), xbmc.LOGERROR)

wiz.refresh()

def autoUpdate(who):

if who == 'all':

for log in DEBRIDID:

if os.path.exists(DEBRIDID[log]['path']):

autoUpdate(log)

elif DEBRIDID[who]:

if os.path.exists(DEBRIDID[who]['path']):

u = debridUser(who)

su = wiz.getS(DEBRIDID[who]['saved'])

n = DEBRIDID[who]['name']

if u == None or u == '': return

elif su == '': debridIt('update', who)

elif not u == su:

if DIALOG.yesno(ADDONTITLE, "[COLOR %s]Would you like to save the [COLOR %s]Real Debrid[/COLOR] data for [COLOR %s]%s[/COLOR]?" % (COLOR2, COLOR1, COLOR1, n), "Addon: [COLOR green][B]%s[/B][/COLOR]" % u, "Saved:[/COLOR] [COLOR red][B]%s[/B][/COLOR]" % su if not su == '' else 'Saved:[/COLOR] [COLOR red][B]None[/B][/COLOR]', yeslabel="[B][COLOR green]Save Data[/COLOR][/B]", nolabel="[B][COLOR red]No Cancel[/COLOR][/B]"):

debridIt('update', who)

else: debridIt('update', who)

def importlist(who):

if who == 'all':

for log in DEBRIDID:

if os.path.exists(DEBRIDID[log]['file']):

importlist(log)

elif DEBRIDID[who]:

if os.path.exists(DEBRIDID[who]['file']):

d = DEBRIDID[who]['default']

sa = DEBRIDID[who]['saved']

su = wiz.getS(sa)

n = DEBRIDID[who]['name']

f = open(DEBRIDID[who]['file'],mode='r'); g = f.read().replace('\n','').replace('\r','').replace('\t',''); f.close();

m = re.compile('<debrid><id>%s</id><value>(.+?)</value></debrid>' % d).findall(g)

if len(m) > 0:

if not m[0] == su:

if DIALOG.yesno(ADDONTITLE, "[COLOR %s]Would you like to import the [COLOR %s]Real Debrid[/COLOR] data for [COLOR %s]%s[/COLOR]?" % (COLOR2, COLOR1, COLOR1, n), "File: [COLOR green][B]%s[/B][/COLOR]" % m[0], "Saved:[/COLOR] [COLOR red][B]%s[/B][/COLOR]" % su if not su == '' else 'Saved:[/COLOR] [COLOR red][B]None[/B][/COLOR]', yeslabel="[B][COLOR green]Save Data[/COLOR][/B]", nolabel="[B][COLOR red]No Cancel[/COLOR][/B]"):

wiz.setS(sa, m[0])

wiz.log('[Import Data] %s: %s' % (who, str(m)), xbmc.LOGNOTICE)

else: wiz.log('[Import Data] Declined Import(%s): %s' % (who, str(m)), xbmc.LOGNOTICE)

else: wiz.log('[Import Data] Duplicate Entry(%s): %s' % (who, str(m))), xbmc.LOGNOTICE

else: wiz.log('[Import Data] No Match(%s): %s' % (who, str(m)), xbmc.LOGNOTICE)

def activateDebrid(who):

if DEBRIDID[who]:

if os.path.exists(DEBRIDID[who]['path']):

act = DEBRIDID[who]['activate']

addonid = wiz.addonId(DEBRIDID[who]['plugin'])

if act == '': addonid.openSettings()

else: url = xbmc.executebuiltin(DEBRIDID[who]['activate'])

else: DIALOG.ok(ADDONTITLE, '%s is not currently installed.' % DEBRIDID[who]['name'])

else:

wiz.refresh()

return

check = 0

while debridUser(who) == None:

if check == 30: break

check += 1

time.sleep(10)

wiz.refresh()