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

# 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 zipfile, xbmcaddon, xbmc, uservar, sys, os, time

import wizard as wiz

KODIV = float(xbmc.getInfoLabel("System.BuildVersion")[:4])

if KODIV > 17:

import zfile as zipfile #FTG mod for Kodi 18

else:

import zipfile

ADDON\_ID = uservar.ADDON\_ID

ADDONTITLE = uservar.ADDONTITLE

COLOR1 = uservar.COLOR1

COLOR2 = uservar.COLOR2

ADDON = wiz.addonId(ADDON\_ID)

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

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

GUISETTINGS = os.path.join(USERDATA, 'guisettings.xml')

KEEPFAVS = wiz.getS('keepfavourites')

KEEPSOURCES = wiz.getS('keepsources')

KEEPPROFILES = wiz.getS('keepprofiles')

KEEPADVANCED = wiz.getS('keepadvanced')

KEEPSUPER = wiz.getS('keepsuper')

KEEPREPOS = wiz.getS('keeprepos')

KEEPWHITELIST = wiz.getS('keepwhitelist')

KODIV = float(xbmc.getInfoLabel("System.BuildVersion")[:4])

LOGFILES = ['xbmc.log', 'xbmc.old.log', 'kodi.log', 'kodi.old.log', 'spmc.log', 'spmc.old.log', 'tvmc.log', 'tvmc.old.log', 'Thumbs.db', '.gitignore', '.DS\_Store']

bad\_files = ['onechannelcache.db', 'saltscache.db', 'saltscache.db-shm', 'saltscache.db-wal', 'saltshd.lite.db', 'saltshd.lite.db-shm', 'saltshd.lite.db-wal', 'queue.db', 'commoncache.db', 'access.log', 'trakt.db', 'video\_cache.db']

def all(\_in, \_out, dp=None, ignore=None, title=None):

if dp: return allWithProgress(\_in, \_out, dp, ignore, title)

else: return allNoProgress(\_in, \_out, ignore)

def allNoProgress(\_in, \_out, ignore):

try:

zin = zipfile.ZipFile(\_in, 'r')

zin.extractall(\_out)

except Exception, e:

print str(e)

return False

return True

def allWithProgress(\_in, \_out, dp, ignore, title):

count = 0; errors = 0; error = ''; update = 0; size = 0; excludes = []

try:

zin = zipfile.ZipFile(\_in, 'r')

except Exception, e:

errors += 1; error += '%s\n' % e

wiz.log('Error Checking Zip: %s' % str(e), xbmc.LOGERROR)

return update, errors, error

whitelist = wiz.whiteList('read')

for item in whitelist:

try: name, id, fold = item

except: pass

excludes.append(fold)

if fold.startswith('pvr'):

wiz.setS('pvrclient', id)

nFiles = float(len(zin.namelist()))

zipsize = wiz.convertSize(sum([item.file\_size for item in zin.infolist()]))

zipit = str(\_in).replace('\\', '/').split('/')

title = title if not title == None else zipit[-1].replace('.zip', '')

for item in zin.infolist():

count += 1; prog = int(count / nFiles \* 100); size += item.file\_size

file = str(item.filename).split('/')

skip = False

line1 = '%s [COLOR %s][B][Errors:%s][/B][/COLOR]' % (title, COLOR2, errors)

line2 = '[COLOR %s][B]File:[/B][/COLOR] [COLOR %s]%s/%s[/COLOR] ' % (COLOR2, COLOR1, count, int(nFiles))

line2 += '[COLOR %s][B]Size:[/B][/COLOR] [COLOR %s]%s/%s[/COLOR]' % (COLOR2, COLOR1, wiz.convertSize(size), zipsize)

line3 = '[COLOR %s]%s[/COLOR]' % (COLOR1, item.filename)

if item.filename == 'userdata/sources.xml' and KEEPSOURCES == 'true': skip = True

elif item.filename == 'userdata/favourites.xml' and KEEPFAVS == 'true': skip = True

elif item.filename == 'userdata/profiles.xml' and KEEPPROFILES == 'true': skip = True

elif item.filename == 'userdata/advancedsettings.xml' and KEEPADVANCED == 'true': skip = True

elif file[0] == 'addons' and file[1] in excludes: skip = True

elif file[0] == 'userdata' and file[1] == 'addon\_data' and file[2] in excludes: skip = True

elif file[-1] in LOGFILES: skip = True

elif file[-1] in bad\_files: skip = True

elif file[-1].endswith('.csv'): skip = True

elif not str(item.filename).find('plugin.program.super.favourites') == -1 and KEEPSUPER == 'true': skip = True

elif not str(item.filename).find(ADDON\_ID) == -1 and ignore == None: skip = True

if skip == True: wiz.log("Skipping: %s" % item.filename, xbmc.LOGNOTICE)

else:

try:

zin.extract(item, \_out)

except Exception, e:

errormsg = "[COLOR %s]File:[/COLOR] [COLOR %s]%s[/COLOR]\n" % (COLOR2, COLOR1, file[-1])

errormsg += "[COLOR %s]Folder:[/COLOR] [COLOR %s]%s[/COLOR]\n" % (COLOR2, COLOR1, (item.filename).replace(file[-1],''))

errormsg += "[COLOR %s]Error:[/COLOR] [COLOR %s]%s[/COLOR]\n\n" % (COLOR2, COLOR1, str(e).replace('\\\\','\\').replace("'%s'" % item.filename, ''))

errors += 1; error += errormsg

wiz.log('Error Extracting: %s(%s)' % (item.filename, str(e)), xbmc.LOGERROR)

pass

dp.update(prog, line1, line2, line3)

if dp.iscanceled(): break

if dp.iscanceled():

dp.close()

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), "[COLOR %s]Extract Cancelled[/COLOR]" % COLOR2)

sys.exit()

return prog, errors, error