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

# Copyright (C) 2015 Surfacingx and Whufclee #

# #

# 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 #

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

# Kodi Logfile Uploader #

# Original Code by Team Kodi #

# By Surfacingx and Whufclee #

# #

# Modified to Support Kodi Forks #

# Added Email Logfile Url Support #

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

import os

import re

import socket

import pyqrcode

from urllib import urlencode

from urllib import FancyURLopener

import urllib2

import urlparse

import urllib

import json

import xbmc

import xbmcgui

import xbmcaddon

import xbmcvfs

import uservar

from resources.libs import wizard as wiz

ADDON\_ID = uservar.ADDON\_ID

ADDONTITLE = uservar.ADDONTITLE

COLOR1 = uservar.COLOR1

COLOR2 = uservar.COLOR2

BUILDERNAME = uservar.BUILDERNAME

ADDON = wiz.addonId(ADDON\_ID)

ADDONVERSION = ADDON.getAddonInfo('version')

DIALOG = xbmcgui.Dialog()

URL = 'https://paste.ubuntu.com/'

EXPIRATION = 2592000

REPLACES = (('//.+?:.+?@', '//USER:PASSWORD@'),('<user>.+?</user>', '<user>USER</user>'),('<pass>.+?</pass>', '<pass>PASSWORD</pass>'),)

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

LOG = xbmc.translatePath('special://logpath/')

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

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

WIZLOG = os.path.join(ADDONDATA, 'wizard.log')

socket.setdefaulttimeout(5)

class QRCode(xbmcgui.WindowXMLDialog):

def \_\_init\_\_(self, \*args, \*\*kwargs):

self.image = kwargs["image"]

self.text = kwargs["text"]

def onInit(self):

self.imagecontrol = 501

self.textbox = 502

self.okbutton = 503

self.title = 504

self.showdialog()

def showdialog(self):

self.getControl(self.imagecontrol).setImage(self.image)

self.getControl(self.textbox).setText(self.text)

self.getControl(self.title).setLabel(ADDONTITLE)

self.setFocus(self.getControl(self.okbutton))

def onClick(self, controlId):

if (controlId == self.okbutton):

self.close()

# Custom urlopener to set user-agent

class pasteURLopener(FancyURLopener):

version = '%s: %s' % (ADDON\_ID, ADDONVERSION)

class Main:

def \_\_init\_\_(self):

self.getSettings()

files = self.getFiles()

for item in files:

filetype = item[0]

if filetype == 'log':

log = wiz.Grab\_Log(file=True).replace(LOG, "")

name = log if log != False else "kodi.log"

error = "Error posting the %s file" % name

elif filetype == 'oldlog':

log = wiz.Grab\_Log(file=True, old=True).replace(LOG, "")

name = log if log != False else "kodi.old.log"

error = "Error posting the %s file" % name

elif filetype == 'wizlog':

name = "wizard.log"

error = "Error posting the %s file" % name

elif filetype == 'crashlog':

name = "crash log"

error = "Error posting the crashlog file"

succes, data = self.readLog(item[1])

if succes:

content = self.cleanLog(data)

succes, result = self.postLog(content, name)

if succes:

msg = "Post this url or scan QRcode for your [COLOR %s]%s[/COLOR], together with a description of the problem:[CR][COLOR %s]%s[/COLOR]" % (COLOR1, name, COLOR1, result)

if len(self.email) > 5:

em\_result, em\_msg = self.email\_Log(self.email, result, name)

if em\_result == 'message':

msg += "[CR]%s" % em\_msg

else:

msg += "[CR]Email ERROR: %s" % em\_msg

self.showResult(msg, result)

else:

self.showResult('%s[CR]%s' % (error, result))

else:

self.showResult('%s[CR]%s' % (error, result))

def getSettings(self):

self.oldlog = ADDON.getSetting('oldlog') == 'true'

self.wizlog = ADDON.getSetting('wizlog') == 'true'

self.crashlog = ADDON.getSetting('crashlog') == 'true'

self.email = ADDON.getSetting('email')

def getFiles(self):

logfiles = []

log = wiz.Grab\_Log(file=True)

old = wiz.Grab\_Log(file=True, old=True)

wizard = False if not os.path.exists(WIZLOG) else WIZLOG

if log != False:

if os.path.exists(log): logfiles.append(['log', log])

else: self.showResult("No log file found")

else: self.showResult("No log file found")

if self.oldlog:

if old != False:

if os.path.exists(old): logfiles.append(['oldlog', old])

else: self.showResult("No old log file found")

else: self.showResult("No old log file found")

if self.wizlog:

if wizard != False:

logfiles.append(['wizlog', wizard])

else: self.showResult("No wizard log file found")

if self.crashlog:

crashlog\_path = ''

items = []

if xbmc.getCondVisibility('system.platform.osx'):

crashlog\_path = os.path.join(os.path.expanduser('~'), 'Library/Logs/DiagnosticReports/')

filematch = 'Kodi'

elif xbmc.getCondVisibility('system.platform.ios'):

crashlog\_path = '/var/mobile/Library/Logs/CrashReporter/'

filematch = 'Kodi'

elif wiz.platform() == 'linux':

crashlog\_path = os.path.expanduser('~') # not 100% accurate (crashlogs can be created in the dir kodi was started from as well)

filematch = 'kodi\_crashlog'

elif wiz.platform() == 'windows':

wiz.log("Windows crashlogs are not supported, please disable this option in the addon settings", xbmc.LOGNOTICE)

#self.showResult("Windows crashlogs are not supported, please disable this option in the addon settings")

elif wiz.platform() == 'android':

wiz.log("Android crashlogs are not supported, please disable this option in the addon settings", xbmc.LOGNOTICE)

#self.showResult("Android crashlogs are not supported, please disable this option in the addon settings")

if crashlog\_path and os.path.isdir(crashlog\_path):

dirs, files = xbmcvfs.listdir(crashlog\_path)

for item in files:

if filematch in item and os.path.isfile(os.path.join(crashlog\_path, item)):

items.append(os.path.join(crashlog\_path, item))

items.sort(key=lambda f: os.path.getmtime(f))

lastcrash = items[-1]

logfiles.append(['crashlog', lastcrash])

if len(items) == 0:

wiz.log("No crashlog file found", xbmc.LOGNOTICE)

return logfiles

def readLog(self, path):

try:

lf = xbmcvfs.File(path)

content = lf.read()

lf.close()

if content:

return True, content

else:

wiz.log('file is empty', xbmc.LOGNOTICE)

return False, "File is Empty"

except:

wiz.log('unable to read file', xbmc.LOGNOTICE)

return False, "Unable to Read File"

def cleanLog(self, content):

for pattern, repl in REPLACES:

content = re.sub(pattern, repl, content)

return content

def postLog(self, data, name):

params = {}

params['poster'] = BUILDERNAME

params['content'] = data

params['syntax'] = 'text'

params['expiration'] = 'week'

params = urlencode(params)

url\_opener = pasteURLopener()

try:

page = url\_opener.open(URL, params)

except Exception, e:

a = 'failed to connect to the server'

wiz.log("%s: %s" % (a, str(e)), xbmc.LOGERROR)

return False, a

try:

page\_url = page.url.strip()

wiz.log("URL for %s: %s" % (name, page\_url), xbmc.LOGNOTICE)

return True, page\_url

except Exception, e:

a = 'unable to retrieve the paste url'

wiz.log("%s: %s" % (a, str(e)), xbmc.LOGERROR)

return False, a

def showResult(self, message, url=None):

if not url == None:

try:

fn = url.split('/')[-2]

imagefile = wiz.generateQR(url, fn)

qr = QRCode( "loguploader.xml" , ADDON.getAddonInfo('path'), 'DefaultSkin', image=imagefile, text=message)

qr.doModal()

del qr

try:

os.remove(imagefile)

except:

pass

except Exception, e:

wiz.log(str(e), xbmc.LOGNOTICE)

confirm = DIALOG.ok(ADDONTITLE, "[COLOR %s]%s[/COLOR]" % (COLOR2, message))

else:

confirm = DIALOG.ok(ADDONTITLE, "[COLOR %s]%s[/COLOR]" % (COLOR2, message))

if ( \_\_name\_\_ == '\_\_main\_\_' ):

Main()