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

# /T /I #

# / |/ | .-~/ #

# T\ Y I |/ / \_ #

# /T | \I | I Y.-~/ #

# I l /I T\ | | l | T / #

# T\ | \ Y l /T | \I l \ ` l Y If your going to copy #

# \_\_ | \l \l \I l \_\_l l \ ` \_. | this addon just #

# \ ~-l `\ `\ \ \ ~\ \ `. .-~ | give credit! #

# \ ~-. "-. ` \ ^.\_ ^. "-. / \ | #

#.--~-.\_ ~- ` \_ ~-\_.-"-." .\_ /.\_ ." ./ Stop Deleting the #

# >--. ~-. .\_ ~>-" "\ 7 7 ] credits file! #

#^.\_\_\_~"--.\_ ~-{ .-~ . `\ Y . / | #

# <\_\_ ~"-. ~ /\_/ \ \I Y : | #

# ^-.\_\_ ~(\_/ \ >.\_: | l\_\_\_\_\_\_ #

# ^--.,\_\_\_.-~" /\_/ ! `-.~"--l\_ / ~"-. #

# (\_/ . ~( /' "~"--,Y -=b-. \_) #

# (\_/ . \ : / l c"~o \ #

# \ / `. . .^ \\_.-~"~--. ) #

# (\_/ . ` / / ! )/ #

# / / \_. '. .': / ' #

# ~(\_/ . / \_ ` .-<\_ #

# /\_/ . ' .-~" `. / \ \ ,z=. Surfacingx #

# ~( / ' : | K "-.~-.\_\_\_\_\_\_// Original Author #

# "-,. l I/ \\_ \_\_{--->.\_(==. #

# //( \ < ~"~" // #

# /' /\ \ \ ,v=. (( Fire TV Guru #

# .^. / /\ " }\_\_ //===- ` PyXBMCt LaYOUt #

# / / ' ' "-.,\_\_ {---(==- #

# .^ ' : T ~" ll #

# / . . . : | :! \ #

# (\_/ / | | j-" ~^ #

# ~-<\_(\_.^-~" #

# #

# Copyright (C) One of those Years.... #

# #

# 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. #

# #

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

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

import shutil

import urllib2,urllib

import re

import zipfile

import uservar

import fnmatch

try: from sqlite3 import dbapi2 as database

except: from pysqlite2 import dbapi2 as database

from datetime import date, datetime, timedelta

from urlparse import urljoin

from resources.libs import extract, downloader, notify, debridit, traktit, allucit, loginit, net, skinSwitch, uploadLog, yt, speedtest, wizard as wiz, addonwindow as pyxbmct

ADDON\_ID = uservar.ADDON\_ID

ADDONTITLE = uservar.ADDONTITLE

ADDON = wiz.addonId(ADDON\_ID)

VERSION = wiz.addonInfo(ADDON\_ID,'version')

ADDONPATH = wiz.addonInfo(ADDON\_ID, 'path')

DIALOG = xbmcgui.Dialog()

DP = xbmcgui.DialogProgress()

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

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

PROFILE = xbmc.translatePath('special://profile/')

TEMPDIR = xbmc.translatePath('special://temp')

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')

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

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

ADVANCED = os.path.join(USERDATA, 'advancedsettings.xml')

SOURCES = os.path.join(USERDATA, 'sources.xml')

FAVOURITES = os.path.join(USERDATA, 'favourites.xml')

PROFILES = os.path.join(USERDATA, 'profiles.xml')

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

THUMBS = os.path.join(USERDATA, 'Thumbnails')

DATABASE = os.path.join(USERDATA, 'Database')

FANART = os.path.join(PLUGIN, 'fanart.jpg')

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

ART = os.path.join(PLUGIN, 'resources', 'art')

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

SPEEDTESTFOLD = os.path.join(ADDONDATA, 'SpeedTest')

ARCHIVE\_CACHE = os.path.join(TEMPDIR, 'archive\_cache')

SKIN = xbmc.getSkinDir()

BUILDNAME = wiz.getS('buildname')

DEFAULTSKIN = wiz.getS('defaultskin')

DEFAULTNAME = wiz.getS('defaultskinname')

DEFAULTIGNORE = wiz.getS('defaultskinignore')

BUILDVERSION = wiz.getS('buildversion')

BUILDTHEME = wiz.getS('buildtheme')

BUILDLATEST = wiz.getS('latestversion')

SHOW15 = wiz.getS('show15')

SHOW16 = wiz.getS('show16')

SHOW17 = wiz.getS('show17')

SHOW18 = wiz.getS('show18')

SHOWADULT = wiz.getS('adult')

SHOWMAINT = wiz.getS('showmaint')

AUTOCLEANUP = wiz.getS('autoclean')

AUTOCACHE = wiz.getS('clearcache')

AUTOPACKAGES = wiz.getS('clearpackages')

AUTOTHUMBS = wiz.getS('clearthumbs')

AUTOFEQ = wiz.getS('autocleanfeq')

AUTONEXTRUN = wiz.getS('nextautocleanup')

INCLUDEVIDEO = wiz.getS('includevideo')

INCLUDEALL = wiz.getS('includeall')

INCLUDEBOB = wiz.getS('includebob')

INCLUDEPHOENIX = wiz.getS('includephoenix')

INCLUDESPECTO = wiz.getS('includespecto')

INCLUDEGENESIS = wiz.getS('includegenesis')

INCLUDEEXODUS = wiz.getS('includeexodus')

INCLUDEONECHAN = wiz.getS('includeonechan')

INCLUDESALTS = wiz.getS('includesalts')

INCLUDESALTSHD = wiz.getS('includesaltslite')

SEPERATE = wiz.getS('seperate')

NOTIFY = wiz.getS('notify')

NOTEID = wiz.getS('noteid')

NOTEDISMISS = wiz.getS('notedismiss')

TRAKTSAVE = wiz.getS('traktlastsave')

REALSAVE = wiz.getS('debridlastsave')

ALLUCSAVE = wiz.getS('alluclastsave')

LOGINSAVE = wiz.getS('loginlastsave')

KEEPFAVS = wiz.getS('keepfavourites')

FAVSsave = wiz.getS('favouriteslastsave')

KEEPSOURCES = wiz.getS('keepsources')

KEEPPROFILES = wiz.getS('keepprofiles')

KEEPADVANCED = wiz.getS('keepadvanced')

KEEPREPOS = wiz.getS('keeprepos')

KEEPSUPER = wiz.getS('keepsuper')

KEEPWHITELIST = wiz.getS('keepwhitelist')

KEEPTRAKT = wiz.getS('keeptrakt')

KEEPREAL = wiz.getS('keepdebrid')

KEEPALLUC = wiz.getS('keepalluc')

KEEPLOGIN = wiz.getS('keeplogin')

DEVELOPER = wiz.getS('developer')

THIRDPARTY = wiz.getS('enable3rd')

THIRD1NAME = wiz.getS('wizard1name')

THIRD1URL = wiz.getS('wizard1url')

THIRD2NAME = wiz.getS('wizard2name')

THIRD2URL = wiz.getS('wizard2url')

THIRD3NAME = wiz.getS('wizard3name')

THIRD3URL = wiz.getS('wizard3url')

BACKUPLOCATION = ADDON.getSetting('path') if not ADDON.getSetting('path') == '' else 'special://home/'

BACKUPROMS = wiz.getS('rompath')

MYBUILDS = os.path.join(BACKUPLOCATION, 'My\_Builds', '')

AUTOFEQ = int(float(AUTOFEQ)) if AUTOFEQ.isdigit() else 0

TODAY = date.today()

TOMORROW = TODAY + timedelta(days=1)

THREEDAYS = TODAY + timedelta(days=3)

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

if KODIV > 17:

from resources.libs import zfile as zipfile #FTG mod for Kodi 18

else:

import zipfile

MCNAME = wiz.mediaCenter()

EXCLUDES = uservar.EXCLUDES

CACHETEXT = uservar.CACHETEXT

CACHEAGE = uservar.CACHEAGE if str(uservar.CACHEAGE).isdigit() else 30

BUILDFILE = uservar.BUILDFILE

ADDONPACK = uservar.ADDONPACK

APKFILE = uservar.APKFILE

YOUTUBETITLE = uservar.YOUTUBETITLE

YOUTUBEFILE = uservar.YOUTUBEFILE

ADDONFILE = uservar.ADDONFILE

ADVANCEDFILE = uservar.ADVANCEDFILE

UPDATECHECK = uservar.UPDATECHECK if str(uservar.UPDATECHECK).isdigit() else 1

NEXTCHECK = TODAY + timedelta(days=UPDATECHECK)

NOTIFICATION = uservar.NOTIFICATION

ENABLE = uservar.ENABLE

HEADERMESSAGE = uservar.HEADERMESSAGE

AUTOUPDATE = uservar.AUTOUPDATE

BUILDERNAME = uservar.BUILDERNAME

WIZARDFILE = uservar.WIZARDFILE

HIDECONTACT = uservar.HIDECONTACT

CONTACT = uservar.CONTACT

CONTACTICON = uservar.CONTACTICON if not uservar.CONTACTICON == 'http://' else ICON

CONTACTFANART = uservar.CONTACTFANART if not uservar.CONTACTFANART == 'http://' else FANART

HIDESPACERS = uservar.HIDESPACERS

COLOR1 = uservar.COLOR1

COLOR2 = uservar.COLOR2

THEME1 = uservar.THEME1

THEME2 = uservar.THEME2

THEME3 = uservar.THEME3

THEME4 = uservar.THEME4

THEME5 = uservar.THEME5

THEME6 = uservar.THEME6

ICONBUILDS = uservar.ICONBUILDS if not uservar.ICONBUILDS == 'http://' else ICON

ICONMAINT = uservar.ICONMAINT if not uservar.ICONMAINT == 'http://' else ICON

ICONAPK = uservar.ICONAPK if not uservar.ICONAPK == 'http://' else ICON

ICONADDONS = uservar.ICONADDONS if not uservar.ICONADDONS == 'http://' else ICON

ICONYOUTUBE = uservar.ICONYOUTUBE if not uservar.ICONYOUTUBE == 'http://' else ICON

ICONSAVE = uservar.ICONSAVE if not uservar.ICONSAVE == 'http://' else ICON

ICONTRAKT = uservar.ICONTRAKT if not uservar.ICONTRAKT == 'http://' else ICON

ICONREAL = uservar.ICONREAL if not uservar.ICONREAL == 'http://' else ICON

ICONLOGIN = uservar.ICONLOGIN if not uservar.ICONLOGIN == 'http://' else ICON

ICONCONTACT = uservar.ICONCONTACT if not uservar.ICONCONTACT == 'http://' else ICON

ICONSETTINGS = uservar.ICONSETTINGS if not uservar.ICONSETTINGS == 'http://' else ICON

Images = xbmc.translatePath(os.path.join('special://home','addons',ADDON\_ID,'resources','images/'));

LOGFILES = wiz.LOGFILES

TRAKTID = traktit.TRAKTID

DEBRIDID = debridit.DEBRIDID

LOGINID = loginit.LOGINID

ALLUCID = allucit.ALLUCID

MODURL = 'http://tribeca.tvaddons.ag/tools/maintenance/modules/'

MODURL2 = 'http://mirrors.kodi.tv/addons/jarvis/'

INSTALLMETHODS = ['Always Ask', 'Reload Profile', 'Force Close']

DEFAULTPLUGINS = ['metadata.album.universal', 'metadata.artists.universal', 'metadata.common.fanart.tv', 'metadata.common.imdb.com', 'metadata.common.musicbrainz.org', 'metadata.themoviedb.org', 'metadata.tvdb.com', 'service.xbmc.versioncheck']

#FTG MOD##

ROMPACK = uservar.ROMPACK

EMUAPKS = uservar.EMUAPKS

ROMPATH = ADDON.getSetting('rompath') if not ADDON.getSetting('rompath') == '' else 'special://home/'

ROMLOC = os.path.join(ROMPATH, 'Roms', '')

try:

INSTALLMETHOD = int(float(wiz.getS('installmethod')))

except:

INSTALLMETHOD = 0

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

###### Menu Items #######

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

#addDir (display,mode,name=None,url=None,menu=None,overwrite=True,fanart=FANART,icon=ICON, themeit=None)

#addFile(display,mode,name=None,url=None,menu=None,overwrite=True,fanart=FANART,icon=ICON, themeit=None)

def index():

errors = int(errorChecking(count=True))

err = str(errors)

errorsfound = '[COLOR red]%s[/COLOR] Error(s) Found' % (err) if errors > 0 else 'None Found'

if AUTOUPDATE == 'Yes':

wizfile = wiz.textCache(WIZARDFILE)

if not wizfile == False:

ver = wiz.checkWizard('version')

if ver > VERSION: addFile('%s [v%s] [COLOR red][B][UPDATE v%s][/B][/COLOR]' % (ADDONTITLE, VERSION, ver), 'wizardupdate', themeit=THEME2)

else: addFile('%s [v%s]' % (ADDONTITLE, VERSION), '', themeit=THEME2)

else: addFile('%s [v%s]' % (ADDONTITLE, VERSION), '', themeit=THEME2)

else: addFile('%s [v%s]' % (ADDONTITLE, VERSION), '', themeit=THEME2)

if len(BUILDNAME) > 0:

version = wiz.checkBuild(BUILDNAME, 'version')

build = '%s (v%s)' % (BUILDNAME, BUILDVERSION)

if version > BUILDVERSION: build = '%s [COLOR red][B][UPDATE v%s][/B][/COLOR]' % (build, version)

addDir(build,'viewbuild',BUILDNAME, themeit=THEME4)

themefile = wiz.themeCount(BUILDNAME)

if not themefile == False:

addFile('None' if BUILDTHEME == "" else BUILDTHEME, 'theme', BUILDNAME, themeit=THEME5)

else: addDir('None', 'builds', themeit=THEME4)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', themeit=THEME3)

addDir ('Builds', 'builds', icon=ICONBUILDS, themeit=THEME1)

addDir ('Maintenance', 'maint', icon=ICONMAINT, themeit=THEME1)

addDir ('Internet Tools' ,'net', icon=ICONCONTACT, themeit=THEME1)

if wiz.platform() == 'android' or DEVELOPER == 'true': addDir ('Apk Installer' ,'apk', icon=ICONAPK, themeit=THEME1)

if wiz.platform() == 'android' or wiz.platform() == 'windows' or DEVELOPER == 'true': addDir ('Retro Gaming Zone' ,'retromenu', icon=ICONSAVE, themeit=THEME1)

if not ADDONFILE == 'http://': addDir ('Addon Installer' ,'addons', icon=ICONADDONS, themeit=THEME1)

if not YOUTUBEFILE == 'http://' and not YOUTUBETITLE == '': addDir (YOUTUBETITLE ,'youtube', icon=ICONYOUTUBE, themeit=THEME1)

addDir ('Save Login Data / Favs Options', 'savedata', icon=ICONSAVE, themeit=THEME1)

addDir ('Backup/Restore Data Options' ,'backup', icon=ICONSAVE, themeit=THEME1)

if HIDECONTACT == 'No': addFile('Contact' ,'contact', icon=ICONCONTACT, themeit=THEME1)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', themeit=THEME3)

addFile('Upload Log File', 'uploadlog', icon=ICONMAINT, themeit=THEME1)

addFile('View Errors in Log: %s' % (errorsfound), 'viewerrorlog', icon=ICONMAINT, themeit=THEME1)

if errors > 0: addFile('View Last Error In Log', 'viewerrorlast', icon=ICONMAINT, themeit=THEME1)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', themeit=THEME3)

addFile('Settings', 'settings', icon=ICONSETTINGS, themeit=THEME1)

addFile('Force Update Text Files', 'forcetext', icon=ICONMAINT, themeit=THEME1)

if DEVELOPER == 'true': addDir('Developer Menu', 'developer', icon=ICON, themeit=THEME1)

setView('files', 'viewType')

def KodiVer():

if KODIV >= 16.0 and KODIV <= 16.9:vername = 'Jarvis'

elif KODIV >= 17.0 and KODIV <= 17.9:vername = 'Krypton'

elif KODIV >= 18.0 and KODIV <= 18.9:vername = 'Leia'

else: vername = "Unknown"

return vername

def buildMenu():

kodi\_ver = KodiVer()

bf = wiz.textCache(BUILDFILE)

if bf == False:

WORKINGURL = wiz.workingURL(BUILDFILE)

addFile('%s Version: %s' % (MCNAME, KODIV), '', icon=ICONBUILDS, themeit=THEME3)

addDir ('Save Data Menu' ,'savedata', icon=ICONSAVE, themeit=THEME3)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', themeit=THEME3)

addFile('Url for txt file not valid', '', icon=ICONBUILDS, themeit=THEME3)

addFile('%s' % WORKINGURL, '', icon=ICONBUILDS, themeit=THEME3)

return

total, count15, count16, count17, count18, adultcount, hidden = wiz.buildCount()

third = False; addin = []

if THIRDPARTY == 'true':

if not THIRD1NAME == '' and not THIRD1URL == '': third = True; addin.append('1')

if not THIRD2NAME == '' and not THIRD2URL == '': third = True; addin.append('2')

if not THIRD3NAME == '' and not THIRD3URL == '': third = True; addin.append('3')

link = bf.replace('\n','').replace('\r','').replace('\t','').replace('gui=""', 'gui="http://"').replace('theme=""', 'theme="http://"').replace('adult=""', 'adult="no"').replace('url2=""', 'url2="http://"').replace('url3=""', 'url3="http://"').replace('preview=""', 'preview="http://"')

match = re.compile('name="(.+?)".+?ersion="(.+?)".+?rl="(.+?)".+?rl2="(.+?)".+?rl3="(.+?)".+?ui="(.+?)".+?odi="(.+?)".+?heme="(.+?)".+?con="(.+?)".+?anart="(.+?)".+?dult="(.+?)".+?escription="(.+?)".+?review="(.+?)"').findall(link)

if total == 1 and third == False:

for name, version, url, url2, url3, gui, kodi, theme, icon, fanart, adult, description, preview in match:

if not SHOWADULT == 'true' and adult.lower() == 'yes': continue

if not DEVELOPER == 'true' and wiz.strTest(name): continue

viewBuild(match[0][0])

return

addFile('%s Version: %s' % (MCNAME, KODIV), '', icon=ICONBUILDS, themeit=THEME3)

addDir ('Save Data Menu' ,'savedata', icon=ICONSAVE, themeit=THEME3)

addDir ('[COLOR yellow]---[B][COLOR lime]Addon Packs [COLOR blue]/ [COLOR red]Fixes[/COLOR][/B][COLOR yellow]---[/COLOR]' ,'viewpack', icon=ICONMAINT, themeit=THEME1)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', themeit=THEME3)

if third == True:

for item in addin:

name = eval('THIRD%sNAME' % item)

addDir ("[B]%s[/B]" % name, 'viewthirdparty', item, icon=ICONBUILDS, themeit=THEME3)

if len(match) >= 1:

if SEPERATE == 'true':

for name, version, url, url2, url3, gui, kodi, theme, icon, fanart, adult, description, preview in match:

if not SHOWADULT == 'true' and adult.lower() == 'yes': continue

if not DEVELOPER == 'true' and wiz.strTest(name): continue

menu = createMenu('install', '', name)

addDir('[%s] %s (v%s)' % (float(kodi), name, version), 'viewbuild', name, description=description, fanart=fanart,icon=icon, menu=menu, themeit=THEME2)

elif DEVELOPER == 'true':

if count15 > 0:

addFile('[B]Test builds[/B]', 'togglesetting', 'show15', themeit=THEME3)

for name, version, url, url2, url3, gui, kodi, theme, icon, fanart, adult, description, preview in match:

if not SHOWADULT == 'true' and adult.lower() == 'yes': continue

if not DEVELOPER == 'true' and wiz.strTest(name): continue

kodiv = int(float(kodi))

if kodiv <= 15:

menu = createMenu('install', '', name)

addDir(' %s (v%s)' % (name, version), 'viewbuild', name, description=description, fanart=fanart,icon=icon, menu=menu, themeit=THEME2)

if count18 > 0:

state = '+' if SHOW18 == 'false' else '-'

addFile('[B]%s Leia Builds(%s)[/B]' % (state, count18), 'togglesetting', 'show18', themeit=THEME3)

if SHOW18 == 'true':

for name, version, url, url2, url3, gui, kodi, theme, icon, fanart, adult, description, preview in match:

if not SHOWADULT == 'true' and adult.lower() == 'yes': continue

if not DEVELOPER == 'true' and wiz.strTest(name): continue

kodiv = int(float(kodi))

if kodiv == 18:

menu = createMenu('install', '', name)

addDir('[%s] %s (v%s)' % (float(kodi), name, version), 'viewbuild', name, description=description, fanart=fanart,icon=icon, menu=menu, themeit=THEME2)

if count17 > 0:

state = '+' if SHOW17 == 'false' else '-'

addFile('[B]%s Krypton Builds(%s)[/B]' % (state, count17), 'togglesetting', 'show17', themeit=THEME3)

if SHOW17 == 'true':

for name, version, url, url2, url3, gui, kodi, theme, icon, fanart, adult, description, preview in match:

if not SHOWADULT == 'true' and adult.lower() == 'yes': continue

if not DEVELOPER == 'true' and wiz.strTest(name): continue

kodiv = int(float(kodi))

if kodiv == 17:

menu = createMenu('install', '', name)

addDir('[%s] %s (v%s)' % (float(kodi), name, version), 'viewbuild', name, description=description, fanart=fanart,icon=icon, menu=menu, themeit=THEME2)

if count16 > 0:

state = '+' if SHOW16 == 'false' else '-'

addFile('[B]%s Jarvis Builds(%s)[/B]' % (state, count16), 'togglesetting', 'show16', themeit=THEME3)

if SHOW16 == 'true':

for name, version, url, url2, url3, gui, kodi, theme, icon, fanart, adult, description, preview in match:

if not SHOWADULT == 'true' and adult.lower() == 'yes': continue

if not DEVELOPER == 'true' and wiz.strTest(name): continue

kodiv = int(float(kodi))

if kodiv == 16:

menu = createMenu('install', '', name)

addDir('[%s] %s (v%s)' % (float(kodi), name, version), 'viewbuild', name, description=description, fanart=fanart,icon=icon, menu=menu, themeit=THEME2)

else:

if kodi\_ver == "Leia":

state = '+' if SHOW18 == 'false' else '-'

addFile('[B]%s Leia Builds(%s)[/B]' % (state, count18), 'togglesetting', 'show18', themeit=THEME3)

if SHOW18 == 'true':

for name, version, url, url2, url3, gui, kodi, theme, icon, fanart, adult, description, preview in match:

if not SHOWADULT == 'true' and adult.lower() == 'yes': continue

if not DEVELOPER == 'true' and wiz.strTest(name): continue

kodiv = int(float(kodi))

if kodiv == 18:

menu = createMenu('install', '', name)

addDir('[%s] %s (v%s)' % (float(kodi), name, version), 'viewbuild', name, description=description, fanart=fanart,icon=icon, menu=menu, themeit=THEME2)

elif kodi\_ver == "Krypton":

state = '+' if SHOW17 == 'false' else '-'

addFile('[B]%s Krypton Builds(%s)[/B]' % (state, count17), 'togglesetting', 'show17', themeit=THEME3)

if SHOW17 == 'true':

for name, version, url, url2, url3, gui, kodi, theme, icon, fanart, adult, description, preview in match:

if not SHOWADULT == 'true' and adult.lower() == 'yes': continue

if not DEVELOPER == 'true' and wiz.strTest(name): continue

kodiv = int(float(kodi))

if kodiv == 17:

menu = createMenu('install', '', name)

addDir('[%s] %s (v%s)' % (float(kodi), name, version), 'viewbuild', name, description=description, fanart=fanart,icon=icon, menu=menu, themeit=THEME2)

elif kodi\_ver == "Jarvis":

state = '+' if SHOW16 == 'false' else '-'

addFile('[B]%s Jarvis Builds(%s)[/B]' % (state, count16), 'togglesetting', 'show16', themeit=THEME3)

if SHOW16 == 'true':

for name, version, url, url2, url3, gui, kodi, theme, icon, fanart, adult, description, preview in match:

if not SHOWADULT == 'true' and adult.lower() == 'yes': continue

if not DEVELOPER == 'true' and wiz.strTest(name): continue

kodiv = int(float(kodi))

if kodiv == 16:

menu = createMenu('install', '', name)

addDir('[%s] %s (v%s)' % (float(kodi), name, version), 'viewbuild', name, description=description, fanart=fanart,icon=icon, menu=menu, themeit=THEME2)

elif hidden > 0:

if adultcount > 0:

addFile('There is currently only Adult builds', '', icon=ICONBUILDS, themeit=THEME3)

addFile('Enable Show Adults in Addon Settings > Misc', '', icon=ICONBUILDS, themeit=THEME3)

else:

addFile('Currently No Builds Offered from %s' % ADDONTITLE, '', icon=ICONBUILDS, themeit=THEME3)

else: addFile('Text file for builds not formated correctly.', '', icon=ICONBUILDS, themeit=THEME3)

setView('files', 'viewType')

def viewBuild(name):

bf = wiz.textCache(BUILDFILE)

if bf == False:

WORKINGURL = wiz.workingURL(BUILDFILE)

addFile('Url for txt file not valid', '', themeit=THEME3)

addFile('%s' % WORKINGURL, '', themeit=THEME3)

return

if wiz.checkBuild(name, 'version') == False:

addFile('Error reading the txt file.', '', themeit=THEME3)

addFile('%s was not found in the builds list.' % name, '', themeit=THEME3)

return

link = bf.replace('\n','').replace('\r','').replace('\t','').replace('gui=""', 'gui="http://"').replace('theme=""', 'theme="http://"').replace('url2=""', 'url2="http://"').replace('url3=""', 'url3="http://"').replace('preview=""', 'preview="http://"').replace('"https://"', 'preview="http://"')

match = re.compile('name="%s".+?ersion="(.+?)".+?rl="(.+?)".+?rl2="(.+?)".+?rl3="(.+?)".+?ui="(.+?)".+?odi="(.+?)".+?heme="(.+?)".+?con="(.+?)".+?anart="(.+?)".+?dult="(.+?)".+?escription="(.+?)".+?review="(.+?)"' % name).findall(link)

for version, url, url2, url3, gui, kodi, themefile, icon, fanart, adult, description, preview in match:

icon = icon

fanart = fanart

build = '%s (v%s)' % (name, version)

if BUILDNAME == name and version > BUILDVERSION:

build = '%s [COLOR red][CURRENT v%s][/COLOR]' % (build, BUILDVERSION)

addFile(build, '', description=description, fanart=fanart, icon=icon, themeit=THEME4)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', themeit=THEME3)

addDir ('Save Data Menu', 'savedata', icon=ICONSAVE, themeit=THEME3)

addFile('Build Information', 'buildinfo', name, description=description, fanart=fanart, icon=icon, themeit=THEME3)

if not preview == "http://": addFile('View Video Preview', 'buildpreview', name, description=description, fanart=fanart, icon=icon, themeit=THEME3)

temp1 = int(float(KODIV)); temp2 = int(float(kodi))

if not temp1 == temp2:

if temp1 == 16 and temp2 <= 15: warning = False

else: warning = True

else: warning = False

if warning == True:

addFile('BUILD DESIGNED FOR KODI VERSION %s [COLOR yellow](INSTALLED: %s)[/COLOR]' % (str(kodi), str(KODIV)), '', fanart=fanart, icon=icon, themeit=THEME6)

addFile(wiz.sep('INSTALL'), '', fanart=fanart, icon=icon, themeit=THEME3)

addFile('Fresh Start then Install' , 'install', name, 'fresh' , description=description, fanart=fanart, icon=icon, themeit=THEME1)

addFile('Standard Install', 'install', name, 'normal' , description=description, fanart=fanart, icon=icon, themeit=THEME1)

if not gui == 'http://': addFile('Apply guiFix' , 'install', name, 'gui' , description=description, fanart=fanart, icon=icon, themeit=THEME1)

if not themefile == 'http://':

themecheck = wiz.textCache(themefile)

if not themecheck == False:

addFile(wiz.sep('THEMES'), '', fanart=fanart, icon=icon, themeit=THEME3)

link = themecheck.replace('\n','').replace('\r','').replace('\t','')

match = re.compile('name="(.+?)".+?rl="(.+?)".+?con="(.+?)".+?anart="(.+?)".+?escription="(.+?)"').findall(link)

for themename, themeurl, themeicon, themefanart, description in match:

themeicon = themeicon if themeicon == 'http://' else icon

themefanart = themefanart if themefanart == 'http://' else fanart

addFile(themename if not themename == BUILDTHEME else "[B]%s (Installed)[/B]" % themename, 'theme', name, themename, description=description, fanart=themefanart, icon=themeicon, themeit=THEME3)

setView('files', 'viewType')

def viewThirdList(number):

name = eval('THIRD%sNAME' % number)

url = eval('THIRD%sURL' % number)

work = wiz.workingURL(url)

if not work == True:

addFile('Url for txt file not valid', '', icon=ICONBUILDS, themeit=THEME3)

addFile('%s' % WORKINGURL, '', icon=ICONBUILDS, themeit=THEME3)

else:

type, buildslist = wiz.thirdParty(url)

addFile("[B]%s[/B]" % name, '', themeit=THEME3)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', themeit=THEME3)

if type:

for name, version, url, kodi, icon, fanart, adult, description in buildslist:

if not SHOWADULT == 'true' and adult.lower() == 'yes': continue

addFile("[%s] %s v%s" % (kodi, name, version), 'installthird', name, url, icon=icon, fanart=fanart, description=description, themeit=THEME2)

else:

for name, url, icon, fanart, description in buildslist:

addFile(name, 'installthird', name, url, icon=icon, fanart=fanart, description=description, themeit=THEME2)

def editThirdParty(number):

name = eval('THIRD%sNAME' % number)

url = eval('THIRD%sURL' % number)

name2 = wiz.getKeyboard(name, 'Enter the Name of the Wizard')

url2 = wiz.getKeyboard(url, 'Enter the URL of the Wizard Text')

wiz.setS('wizard%sname' % number, name2)

wiz.setS('wizard%surl' % number, url2)

def apkScraper(name=""):

if name == 'kodi':

kodiurl1 = 'http://mirrors.kodi.tv/releases/android/arm/'

kodiurl2 = 'http://mirrors.kodi.tv/releases/android/arm/old/'

url1 = wiz.openURL(kodiurl1).replace('\n', '').replace('\r', '').replace('\t', '')

url2 = wiz.openURL(kodiurl2).replace('\n', '').replace('\r', '').replace('\t', '')

x = 0

match1 = re.compile('<tr><td><a href="(.+?)".+?>(.+?)</a></td><td>(.+?)</td><td>(.+?)</td></tr>').findall(url1)

match2 = re.compile('<tr><td><a href="(.+?)".+?>(.+?)</a></td><td>(.+?)</td><td>(.+?)</td></tr>').findall(url2)

addFile("Official Kodi Apk\'s", themeit=THEME1)

rc = False

for url, name, size, date in match1:

if url in ['../', 'old/']: continue

if not url.endswith('.apk'): continue

if not url.find('\_') == -1 and rc == True: continue

try:

tempname = name.split('-')

if not url.find('\_') == -1:

rc = True

name2, v2 = tempname[2].split('\_')

else:

name2 = tempname[2]

v2 = ''

title = "[COLOR %s]%s v%s%s %s[/COLOR] [COLOR %s]%s[/COLOR] [COLOR %s]%s[/COLOR]" % (COLOR1, tempname[0].title(), tempname[1], v2.upper(), name2, COLOR2, size.replace(' ', ''), COLOR1, date)

download = urljoin(kodiurl1, url)

addFile(title, 'apkinstall', "%s v%s%s %s" % (tempname[0].title(), tempname[1], v2.upper(), name2), download)

x += 1

except:

wiz.log("Error on: %s" % name)

for url, name, size, date in match2:

if url in ['../', 'old/']: continue

if not url.endswith('.apk'): continue

if not url.find('\_') == -1: continue

try:

tempname = name.split('-')

title = "[COLOR %s]%s v%s %s[/COLOR] [COLOR %s]%s[/COLOR] [COLOR %s]%s[/COLOR]" % (COLOR1, tempname[0].title(), tempname[1], tempname[2], COLOR2, size.replace(' ', ''), COLOR1, date)

download = urljoin(kodiurl2, url)

addFile(title, 'apkinstall', "%s v%s %s" % (tempname[0].title(), tempname[1], tempname[2]), download)

x += 1

except:

wiz.log("Error on: %s" % name)

if x == 0: addFile("Error Kodi Scraper Is Currently Down.")

elif name == 'spmc':

spmcurl1 = 'https://github.com/koying/SPMC/releases'

url1 = wiz.openURL(spmcurl1).replace('\n', '').replace('\r', '').replace('\t', '')

x = 0

match1 = re.compile('<div.+?lass="release-body.+?div class="release-header".+?a href=.+?>(.+?)</a>.+?ul class="release-downloads">(.+?)</ul>.+?/div>').findall(url1)

addFile("Official SPMC Apk\'s", themeit=THEME1)

for name, urls in match1:

tempurl = ''

match2 = re.compile('<li>.+?<a href="(.+?)" rel="nofollow">.+?<small class="text-gray float-right">(.+?)</small>.+?strong>(.+?)</strong>.+?</a>.+?</li>').findall(urls)

for apkurl, apksize, apkname in match2:

if apkname.find('armeabi') == -1: continue

if apkname.find('launcher') > -1: continue

tempurl = urljoin('https://github.com', apkurl)

break

if tempurl == '': continue

try:

name = "SPMC %s" % name

title = "[COLOR %s]%s[/COLOR] [COLOR %s]%s[/COLOR]" % (COLOR1, name, COLOR2, apksize.replace(' ', ''))

download = tempurl

addFile(title, 'apkinstall', name, download)

x += 1

except Exception, e:

wiz.log("Error on: %s / %s" % (name, str(e)))

if x == 0: addFile("Error SPMC Scraper Is Currently Down.")

def apkMenu(name=None, url=None):

if HIDESPACERS == 'No': addFile(wiz.sep('Apps from apkfiles.com'), '', themeit=THEME3)

addDir ('App Lists' ,'apkfiles', icon=ICONSAVE, themeit=THEME1)

if HIDESPACERS == 'No': addFile(wiz.sep('FTG Modded Apps'), '', themeit=THEME3)

addDir ('App Lists' ,'ftgmod', icon=ICONSAVE, themeit=THEME1)

setView('files', 'viewType')

if url == None:

if HIDESPACERS == 'No': addFile(wiz.sep('Official Kodi/SPMC'), '', themeit=THEME3)

addDir ('Kodi Apk\'s', 'apkscrape', 'kodi', icon=ICONAPK, themeit=THEME1)

addDir ('SPMC Apk\'s', 'apkscrape', 'spmc', icon=ICONAPK, themeit=THEME1)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', themeit=THEME3)

if not APKFILE == 'http://':

if url == None:

TEMPAPKFILE = wiz.textCache(uservar.APKFILE)

if TEMPAPKFILE == False: APKWORKING = wiz.workingURL(uservar.APKFILE)

else:

TEMPAPKFILE = wiz.textCache(url)

if TEMPAPKFILE == False: APKWORKING = wiz.workingURL(url)

if not TEMPAPKFILE == False:

link = TEMPAPKFILE.replace('\n','').replace('\r','').replace('\t','')

match = re.compile('name="(.+?)".+?ection="(.+?)".+?rl="(.+?)".+?con="(.+?)".+?anart="(.+?)".+?dult="(.+?)".+?escription="(.+?)"').findall(link)

if len(match) > 0:

x = 0

for aname, section, url, icon, fanart, adult, description in match:

if not SHOWADULT == 'true' and adult.lower() == 'yes': continue

if section.lower() == 'yes':

x += 1

addDir ("[B]%s[/B]" % aname, 'apk', aname, url, description=description, icon=icon, fanart=fanart, themeit=THEME3)

elif section.lower() == 'yes':

x += 1

addFile(aname, 'rominstall', aname, url, description=description, icon=icon, fanart=fanart, themeit=THEME2)

else:

x += 1

addFile(aname, 'apkinstall', aname, url, description=description, icon=icon, fanart=fanart, themeit=THEME2)

if x == 0:

addFile("No addons added to this menu yet!", '', themeit=THEME2)

else: wiz.log("[APK Menu] ERROR: Invalid Format.", xbmc.LOGERROR)

else:

wiz.log("[APK Menu] ERROR: URL for apk list not working.", xbmc.LOGERROR)

addFile('Url for txt file not valid', '', themeit=THEME3)

addFile('%s' % APKWORKING, '', themeit=THEME3)

return

else: wiz.log("[APK Menu] No APK list added.")

setView('files', 'viewType')

def ftgmod():

if not APKFILE == 'http://':

APKWORKING = wiz.workingURL(APKFILE)

if APKWORKING == True:

link = wiz.openURL(APKFILE).replace('\n','').replace('\r','').replace('\t','')

match = re.compile('name="(.+?)".+?rl="(.+?)".+?con="(.+?)".+?anart="(.+?)"').findall(link)

if len(match) > 0:

for name, url, icon, fanart in match:

#if not SHOWADULT == 'true' and adult.lower() == 'yes': continue

addDir(name, 'GetList', name, url, icon=icon, fanart=fanart, themeit=THEME1)

else: wiz.log("[APK Menu] ERROR: Invalid Format.")

else:

wiz.log("[APK Menu] ERROR: URL for apk list not working.", xbmc.LOGERROR)

addFile('Url for txt file not valid', '', themeit=THEME3)

addFile('%s' % APKWORKING, '', themeit=THEME3)

return

else: wiz.log("[APK Menu] No APK list added.")

def GetList(url):

if not wiz.workingURL(url) == True: return False

link = wiz.openURL(url).replace('\n','').replace('\r','').replace('\t','')

match = re.compile('name="(.+?)".+?rl="(.+?)".+?con="(.+?)".+?anart="(.+?)"').findall(link)

if len(match) > 0:

for name, url, icon, fanart in match:

addFile(name, 'apkinstall', name, url, icon=icon, fanart=fanart, themeit=THEME1)

else: wiz.log("[APK Menu] ERROR: Invalid Format.")

else: wiz.log("[APK Menu] ERROR: URL for emu list not working.")

def apkfiles():

if HIDESPACERS == 'No': addFile(wiz.sep('Apps from apkfiles.com'), '', themeit=THEME3)

html= wiz.openURL('https://www.apkfiles.com/')

match = re.compile('href="([^"]\*)">Applications(.+?)</a>').findall(html)

match2 = re.compile('href="([^"]\*)">Games(.+?)</a>').findall(html)

for url,count in match:

addDir2('[COLOR blue]Android Apps[/COLOR]','https://www.apkfiles.com' +url,'apkgame',ICONAPK,FANART)

for url,count in match2:

addDir2('[COLOR blue]Android Games[/COLOR]','https://www.apkfiles.com' +url,'apkgame',ICONAPK,FANART)

setView('movies', 'MAIN')

def apkshowMenu(url):

if not wiz.workingURL(url) == True: return False

link = wiz.openURL(url).replace('\n','').replace('\r','').replace('\t','')

match = re.compile('name="(.+?)".+?rl="(.+?)".+?con="(.+?)".+?anart="(.+?)"').findall(link)

if len(match) > 0:

for name, url, icon, fanart in match:

addFile(name, 'apkinstall', name, url, icon=icon, fanart=fanart, themeit=THEME1)

else: wiz.log("[APK Menu] ERROR: Invalid Format.")

else: wiz.log("[APK Menu] ERROR: URL for emu list not working.")

def APKGAME(url):

html=wiz.openURL(url)

match = re.compile('<a href="([^"]\*)" >(.+?)</a>').findall(html)

for url,name in match:

if '/cat' in url:

addDir2((name).replace('&amp;',' - '),'https://www.apkfiles.com'+url,'select',ART+'APK.png',FANART)

def APKSELECT2(url):

html=wiz.openURL(url)

url1 = url

if "page=" in str(url):

url1 = url.split('?')[0]

match = re.compile('<a href="([^"]\*)".+?<img src="([^"]\*)" class="file\_list\_icon".+?alt="([^"]\*)"',re.DOTALL).findall(html)

match2 = re.compile('class="[^"]\*".+?ref="([^"]\*)".+?yle=.+?</a>').findall(html)

for url,img,name in match:

if 'apk' in url:

addDir2((name).replace('&#39;','').replace('&amp;',' - ').replace('&#174;:',': ').replace('&#174;',' '),'https://www.apkfiles.com'+url,'grab','http:'+img, FANART)

if len(match2) > 1:

match2 = str(match2[len(match2) - 1])

addDir2('Next Page',url1+str(match2),'select',ART+'Next.png',FANART)

def APKGRAB(name,url):

html=wiz.openURL(url)

name=name

match = re.compile('href="([^"]\*)".+?lass="yellow\_button".+?itle=').findall(html)

for url in match:

url = 'https://www.apkfiles.com'+url

apkInstaller1(name,url)

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

#################################RETRO PACKS###############################

def retromenu():

MKDIRS()#if not os.path.exists(ROMLOC): os.makedirs(ROMLOC)

if HIDESPACERS == 'No': addFile(wiz.sep('Emulators'), '', themeit=THEME3)

if wiz.platform() == 'android' or DEVELOPER == 'true': addDir ('Emulator APKs' ,'emumenu', icon=ICONSAVE, themeit=THEME1)

if wiz.platform() == 'windows' or DEVELOPER == 'true': addDir ('Emulator APPs' ,'emumenu', icon=ICONSAVE, themeit=THEME1)

if HIDESPACERS == 'No': addFile(wiz.sep('Rom Packs'), '', themeit=THEME3)

addDir ('Rom Pack Zips' ,'rompackmenu', icon=ICONSAVE, themeit=THEME1)

def emumenu():

link = wiz.openURL(EMUAPKS).replace('\n','').replace('\r','').replace('\t','')

match = re.compile('name="(.+?)".+?rl="(.+?)".+?con="(.+?)".+?anart="(.+?)"').findall(link)

if len(match) > 0:

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

for name, url, icon, fanart in match:

addFile(name, 'apkinstall', name, url, icon=icon, fanart=fanart, themeit=THEME1)

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

DIALOG.ok(ADDONTITLE, "[COLOR yellow]Please go download RetroArch for PC[/COLOR]", " Goto http://tinyurl.com/RetroFTG for a full tutorial")

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

DIALOG.ok(ADDONTITLE, "[COLOR yellow]Please go download RetroArch for PC[/COLOR]", " Goto http://tinyurl.com/RetroFTG for a full tutorial")

def rompackmenu():

link = wiz.openURL(ROMPACK).replace('\n','').replace('\r','').replace('\t','')

match = re.compile('name="(.+?)".+?rl="(.+?)".+?con="(.+?)".+?anart="(.+?)"').findall(link)

if len(match) > 0:

for name, url, icon, fanart in match:

addFile(name, 'UNZIPROM', name, url, icon=icon, fanart=fanart, themeit=THEME1)

def UNZIPROM():

myroms = xbmc.translatePath(BACKUPROMS)

if myroms == '':

if DIALOG.yesno(ADDONTITLE, "[COLOR %s]Parece que voce nao tem um local de armazenamento das ROOMS" % COLOR2, "Voce gostaria de definir um agora?[/COLOR]", yeslabel="[COLOR green][B]Definir Local[/B][/COLOR]", nolabel="[COLOR red][B]Cancelar Download[/B][/COLOR]"):

wiz.openS('rompath')

myroms = wiz.getS('rompath')

if myroms == '': return

yes = DIALOG.yesno(ADDONTITLE, "[COLOR %s]Tem certeza que voce gostaria de baixar e extrair? [COLOR %s]%s[/COLOR] para:" % (COLOR2, COLOR1, name), "[COLOR %s]%s[/COLOR]" % (COLOR1, myroms), yeslabel="[B][COLOR green]Download[/COLOR][/B]", nolabel="[B][COLOR red]Cancelar[/COLOR][/B]")

if not yes: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]ERROR: Install Cancelled[/COLOR]' % COLOR2); return

display = name

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

if not wiz.workingURL(url) == True: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]Rom Installer: Invalid Rom Url![/COLOR]' % COLOR2); return

DP.create(ADDONTITLE,'[COLOR %s][B]Baixando:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, display),'', 'Por favor aguarde')

lib=os.path.join(PACKAGES, "%s.zip" % name.replace('\\', '').replace('/', '').replace(':', '').replace('\*', '').replace('?', '').replace('"', '').replace('<', '').replace('>', '').replace('|', ''))

try: os.remove(lib)

except: pass

downloader.download(url, lib, DP)

xbmc.sleep(100)

percent, errors, error = extract.all(lib,myroms,DP, title=display)

try: os.remove(lib)

except: pass

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]Rom Pack Installed[/COLOR]' % COLOR2)

DP.close()

def MKDIRS():

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

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

def buildVideo(name):

if wiz.workingURL(BUILDFILE) == True:

videofile = wiz.checkBuild(name, 'preview')

wiz.FTGlog('Name %s'%name)

wiz.FTGlog('URL %s'%videofile)

if videofile and not videofile == 'http://': playVideoB(videofile)

else: wiz.log("[%s]Unable to find url for video preview" % name)

else: wiz.log("Build text file not working: %s" % WORKINGURL)

def playVideo(url):

if 'watch?v=' in url:

a, b = url.split('?')

find = b.split('&')

for item in find:

if item.startswith('v='):

url = item[2:]

break

else: continue

elif 'embed' in url or 'youtu.be' in url:

a = url.split('/')

if len(a[-1]) > 5:

url = a[-1]

elif len(a[-2]) > 5:

url = a[-2]

wiz.log("YouTube URL: %s" % url)

if wiz.getCond('System.HasAddon(plugin.video.youtube)') == 1:

url = 'plugin://plugin.video.youtube/play/?video\_id=%s' % url

xbmc.Player().play(url)

xbmc.sleep(2000)

if xbmc.Player().isPlayingVideo() == 0:

yt.PlayVideo(url)

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

def addonMenu(name=None, url=None):

if not ADDONFILE == 'http://':

if url == None:

TEMPADDONFILE = wiz.textCache(uservar.ADDONFILE)

if TEMPADDONFILE == False: ADDONWORKING = wiz.workingURL(uservar.ADDONFILE)

else:

TEMPADDONFILE = wiz.textCache(url)

if TEMPADDONFILE == False: ADDONWORKING = wiz.workingURL(url)

if not TEMPADDONFILE == False:

link = TEMPADDONFILE.replace('\n','').replace('\r','').replace('\t','').replace('repository=""', 'repository="none"').replace('repositoryurl=""', 'repositoryurl="http://"').replace('repositoryxml=""', 'repositoryxml="http://"')

match = re.compile('name="(.+?)".+?lugin="(.+?)".+?rl="(.+?)".+?epository="(.+?)".+?epositoryxml="(.+?)".+?epositoryurl="(.+?)".+?con="(.+?)".+?anart="(.+?)".+?dult="(.+?)".+?escription="(.+?)"').findall(link)

if len(match) > 0:

x = 0

for aname, plugin, aurl, repository, repositoryxml, repositoryurl, icon, fanart, adult, description in match:

if plugin.lower() == 'section':

addonMenu(name, url)

elif plugin.lower() == 'skin':

skinInstaller(name, url)

elif plugin.lower() == 'pack':

packInstaller(name, url)

else:

if not SHOWADULT == 'true' and adult.lower() == 'yes': continue

try:

add = xbmcaddon.Addon(id=plugin).getAddonInfo('path')

if os.path.exists(add):

aname = "[COLOR green][Installed][/COLOR] %s" % aname

except:

pass

addonInstaller(plugin, url)

if x < 1:

wiz.LogNotify("[COLOR %s]No Addons[/COLOR]" % COLOR1)

else:

addFile('Text File not formated correctly!', '', themeit=THEME3)

wiz.log("[Addon Menu] ERROR: Invalid Format.")

else:

wiz.log("[Addon Menu] ERROR: URL for Addon list not working.")

addFile('Url for txt file not valid', '', themeit=THEME3)

addFile('%s' % ADDONWORKING, '', themeit=THEME3)

else: wiz.log("[Addon Menu] No Addon list added.")

#else:

# wiz.LogNotify("[COLOR %s]error[/COLOR]" % COLOR1)

def packInstaller(name, url):

if not wiz.workingURL(url) == True: wiz.LogNotify("[COLOR %s]Addon Installer[/COLOR]" % COLOR1, '[COLOR %s]%s:[/COLOR] [COLOR %s]URL do arquivo ZIP invalido![/COLOR]' % (COLOR1, name, COLOR2)); return

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

DP.create(ADDONTITLE,'[COLOR %s][B]Baixando:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, name), '', '[COLOR %s]Por favor, agaurde...[/COLOR]' % COLOR2)

urlsplits = url.split('/')

lib = xbmc.makeLegalFilename(os.path.join(PACKAGES, urlsplits[-1]))

try: os.remove(lib)

except: pass

downloader.download(url, lib, DP)

title = '[COLOR %s][B]Instalando:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, name)

DP.update(0, title,'', '[COLOR %s]Por favor, aguarde...[/COLOR]' % COLOR2)

percent, errors, error = extract.all(lib,ADDONS,DP, title=title)

installed = grabAddons(lib)

if KODIV >= 17: wiz.addonDatabase(installed, 1, True)

DP.close()

wiz.LogNotify("[COLOR %s]Addon Installer[/COLOR]" % COLOR1, '[COLOR %s]%s: Installed![/COLOR]' % (COLOR2, name))

wiz.ebi('UpdateAddonRepos()')

wiz.ebi('UpdateLocalAddons()')

wiz.refresh()

def skinInstaller(name, url):

if not wiz.workingURL(url) == True: wiz.LogNotify("[COLOR %s]Addon Installer[/COLOR]" % COLOR1, '[COLOR %s]%s:[/COLOR] [COLOR %s]Invalid Zip Url![/COLOR]' % (COLOR1, name, COLOR2)); return

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

DP.create(ADDONTITLE,'[COLOR %s][B]Baixando:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, name), '', '[COLOR %s]Por favor, aguarde..[/COLOR]' % COLOR2)

urlsplits = url.split('/')

lib = xbmc.makeLegalFilename(os.path.join(PACKAGES, urlsplits[-1]))

try: os.remove(lib)

except: pass

downloader.download(url, lib, DP)

title = '[COLOR %s][B]Instalando:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, name)

DP.update(0, title,'', '[COLOR %s]Por favor, aguarde...[/COLOR]' % COLOR2)

percent, errors, error = extract.all(lib,HOME,DP, title=title)

installed = grabAddons(lib)

if KODIV >= 17: wiz.addonDatabase(installed, 1, True)

DP.close()

wiz.LogNotify("[COLOR %s]Addon Installer[/COLOR]" % COLOR1, '[COLOR %s]%s: Installed![/COLOR]' % (COLOR2, name))

wiz.ebi('UpdateAddonRepos()')

wiz.ebi('UpdateLocalAddons()')

for item in installed:

if item.startswith('skin.') == True and not item == 'skin.shortcuts':

if not BUILDNAME == '' and DEFAULTIGNORE == 'true': wiz.setS('defaultskinignore', 'true')

wiz.swapSkins(item, 'Skin Installer')

wiz.refresh()

def addonInstaller(plugin, url):

if not ADDONFILE == 'http://' or '':

url = ADDONFILE

ADDONWORKING = wiz.workingURL(url)

if ADDONWORKING == True:

link = wiz.textCache(url).replace('\n','').replace('\r','').replace('\t','').replace('repository=""', 'repository="none"').replace('repositoryurl=""', 'repositoryurl="http://"').replace('repositoryxml=""', 'repositoryxml="http://"')

match = re.compile('name="(.+?)".+?lugin="%s".+?rl="(.+?)".+?epository="(.+?)".+?epositoryxml="(.+?)".+?epositoryurl="(.+?)".+?con="(.+?)".+?anart="(.+?)".+?dult="(.+?)".+?escription="(.+?)"' % plugin).findall(link)

if len(match) > 0:

for name, url, repository, repositoryxml, repositoryurl, icon, fanart, adult, description in match:

if os.path.exists(os.path.join(ADDONS, plugin)):

do = ['Iiniciar ADDON', 'Remover ADDON']

selected = DIALOG.select("[COLOR %s]Addons ja esta instalado, o que voce gostaria de fazer?[/COLOR]" % COLOR2, do)

if selected == 0:

wiz.ebi('RunAddon(%s)' % plugin)

xbmc.sleep(500)

return True

elif selected == 1:

wiz.cleanHouse(os.path.join(ADDONS, plugin))

try: wiz.removeFolder(os.path.join(ADDONS, plugin))

except: pass

if DIALOG.yesno(ADDONTITLE, "[COLOR %s]Would you like to remove the addon\_data for:" % COLOR2, "[COLOR %s]%s[/COLOR]?[/COLOR]" % (COLOR1, plugin), yeslabel="[B][COLOR green]Yes Remove[/COLOR][/B]", nolabel="[B][COLOR red]No Skip[/COLOR][/B]"):

removeAddonData(plugin)

wiz.refresh()

return True

else:

return False

repo = os.path.join(ADDONS, repository)

if not repository.lower() == 'none' and not os.path.exists(repo):

wiz.log("Repository not installed, installing it")

if DIALOG.yesno(ADDONTITLE, "[COLOR %s]Would you like to install the repository for [COLOR %s]%s[/COLOR]:" % (COLOR2, COLOR1, plugin), "[COLOR %s]%s[/COLOR]?[/COLOR]" % (COLOR1, repository), yeslabel="[B][COLOR green]Yes Install[/COLOR][/B]", nolabel="[B][COLOR red]No Skip[/COLOR][/B]"):

ver = wiz.parseDOM(wiz.openURL(repositoryxml), 'addon', ret='version', attrs = {'id': repository})

if len(ver) > 0:

repozip = '%s%s-%s.zip' % (repositoryurl, repository, ver[0])

wiz.log(repozip)

if KODIV >= 17: wiz.addonDatabase(repository, 1)

installAddon(repository, repozip)

wiz.ebi('UpdateAddonRepos()')

#wiz.ebi('UpdateLocalAddons()')

wiz.log("Installing Addon from Kodi")

install = installFromKodi(plugin)

wiz.log("Install from Kodi: %s" % install)

if install:

wiz.refresh()

return True

else:

wiz.log("[Addon Installer] Repository not installed: Unable to grab url! (%s)" % repository)

else: wiz.log("[Addon Installer] Repository for %s not installed: %s" % (plugin, repository))

elif repository.lower() == 'none':

wiz.log("No repository, installing addon")

pluginid = plugin

zipurl = url

installAddon(plugin, url)

wiz.refresh()

return True

else:

wiz.log("Repository installed, installing addon")

install = installFromKodi(plugin, False)

if install:

wiz.refresh()

return True

if os.path.exists(os.path.join(ADDONS, plugin)): return True

ver2 = wiz.parseDOM(wiz.openURL(repositoryxml), 'addon', ret='version', attrs = {'id': plugin})

if len(ver2) > 0:

url = "%s%s-%s.zip" % (url, plugin, ver2[0])

wiz.log(str(url))

if KODIV >= 17: wiz.addonDatabase(plugin, 1)

installAddon(plugin, url)

wiz.refresh()

else:

wiz.log("no match"); return False

else: wiz.log("[Addon Installer] Invalid Format")

else: wiz.log("[Addon Installer] Text File: %s" % ADDONWORKING)

else: wiz.log("[Addon Installer] Not Enabled.")

def installFromKodi(plugin, over=True):

if over == True:

xbmc.sleep(2000)

#wiz.ebi('InstallAddon(%s)' % plugin)

wiz.ebi('RunPlugin(plugin://%s)' % plugin)

if not wiz.whileWindow('yesnodialog'):

return False

xbmc.sleep(500)

if wiz.whileWindow('okdialog'):

return False

wiz.whileWindow('progressdialog')

if os.path.exists(os.path.join(ADDONS, plugin)): return True

else: return False

def installAddon(name, url):

if not wiz.workingURL(url) == True: wiz.LogNotify("[COLOR %s]Addon Installer[/COLOR]" % COLOR1, '[COLOR %s]%s:[/COLOR] [COLOR %s]Invalid Zip Url![/COLOR]' % (COLOR1, name, COLOR2)); return

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

DP.create(ADDONTITLE,'[COLOR %s][B]Baixando:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, name), '', '[COLOR %s]Por favor, aguarde..[/COLOR]' % COLOR2)

urlsplits = url.split('/')

lib=os.path.join(PACKAGES, urlsplits[-1])

try: os.remove(lib)

except: pass

downloader.download(url, lib, DP)

title = '[COLOR %s][B]Instalando:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, name)

DP.update(0, title,'', '[COLOR %s]Por favor, aguarde...[/COLOR]' % COLOR2)

percent, errors, error = extract.all(lib,ADDONS,DP, title=title)

DP.update(0, title,'', '[COLOR %s]Instalando dependencias[/COLOR]' % COLOR2)

installed(name)

installlist = grabAddons(lib)

wiz.log(str(installlist))

if KODIV >= 17: wiz.addonDatabase(installlist, 1, True)

installDep(name, DP)

DP.close()

wiz.ebi('UpdateAddonRepos()')

wiz.ebi('UpdateLocalAddons()')

wiz.refresh()

for item in installlist:

if item.startswith('skin.') == True and not item == 'skin.shortcuts':

if not BUILDNAME == '' and DEFAULTIGNORE == 'true': wiz.setS('defaultskinignore', 'true')

wiz.swapSkins(item, 'Skin Installer')

def installDep(name, DP=None):

dep=os.path.join(ADDONS,name,'addon.xml')

if os.path.exists(dep):

source = open(dep,mode='r'); link = source.read(); source.close();

match = wiz.parseDOM(link, 'import', ret='addon')

for depends in match:

if not 'xbmc.python' in depends:

if not DP == None:

DP.update(0, '', '[COLOR %s]%s[/COLOR]' % (COLOR1, depends))

try:

add = xbmcaddon.Addon(id=depends)

name2 = add.getAddonInfo('name')

except:

wiz.createTemp(depends)

if KODIV >= 17: wiz.addonDatabase(depends, 1)

def installed(addon):

url = os.path.join(ADDONS,addon,'addon.xml')

if os.path.exists(url):

try:

list = open(url,mode='r'); g = list.read(); list.close()

name = wiz.parseDOM(g, 'addon', ret='name', attrs = {'id': addon})

icon = os.path.join(ADDONS,addon,'icon.png')

wiz.LogNotify('[COLOR %s]%s[/COLOR]' % (COLOR1, name[0]), '[COLOR %s]Addon Enabled[/COLOR]' % COLOR2, '2000', icon)

except: pass

def youtubeMenu(name=None, url=None):

if not YOUTUBEFILE == 'http://':

if url == None:

TEMPYOUTUBEFILE = wiz.textCache(uservar.YOUTUBEFILE)

if TEMPYOUTUBEFILE == False: YOUTUBEWORKING = wiz.workingURL(uservar.YOUTUBEFILE)

else:

TEMPYOUTUBEFILE = wiz.textCache(url)

if TEMPYOUTUBEFILE == False: YOUTUBEWORKING = wiz.workingURL(url)

if not TEMPYOUTUBEFILE == False:

link = TEMPYOUTUBEFILE.replace('\n','').replace('\r','').replace('\t','')

match = re.compile('name="(.+?)".+?rl="(.+?)".+?con="(.+?)".+?anart="(.+?)".+?escription="(.+?)"').findall(link)

if len(match) > 0:

for name, url, icon, fanart, description in match:

addFile(name, 'viewVideo', url=url, description=description, icon=icon, fanart=fanart, themeit=THEME2)

else: wiz.log("[YouTube Menu] ERROR: Invalid Format.")

else:

wiz.log("[YouTube Menu] ERROR: URL for YouTube list not working.")

addFile('Url for txt file not valid', '', themeit=THEME3)

addFile('%s' % YOUTUBEWORKING, '', themeit=THEME3)

else: wiz.log("[YouTube Menu] No YouTube list added.")

setView('files', 'viewType')

def maintMenu(view=None):

on = '[B][COLOR green]ON[/COLOR][/B]'; off = '[B][COLOR red]OFF[/COLOR][/B]'

if wiz.Grab\_Log(True) == False: kodilog = 0

else: kodilog = errorChecking(wiz.Grab\_Log(True), True)

if wiz.Grab\_Log(True, True) == False: kodioldlog = 0

else: kodioldlog = errorChecking(wiz.Grab\_Log(True,True), True)

errorsinlog = int(kodilog) + int(kodioldlog)

wizlogsize = ': [COLOR red]Not Found[/COLOR]' if not os.path.exists(WIZLOG) else ": [COLOR green]%s[/COLOR]" % wiz.convertSize(os.path.getsize(WIZLOG))

sizepack = wiz.getSize(PACKAGES)

sizethumb = wiz.getSize(THUMBS)

sizecache = wiz.getCacheSize()

totalsize = sizepack+sizethumb+sizecache

addFile('Total Clean Up: [COLOR green][B]%s[/B][/COLOR]' % wiz.convertSize(totalsize) ,'fullclean', icon=ICONMAINT, themeit=THEME3)

addFile('Clear Cache: [COLOR green][B]%s[/B][/COLOR]' % wiz.convertSize(sizecache) ,'clearcache', icon=ICONMAINT, themeit=THEME3)

addFile('Limpando Temporarios: [COLOR green][B]%s[/B][/COLOR]' % wiz.convertSize(sizepack) ,'clearpackages', icon=ICONMAINT, themeit=THEME3)

addFile('Clear Thumbnails: [COLOR green][B]%s[/B][/COLOR]' % wiz.convertSize(sizethumb),'clearthumb', icon=ICONMAINT, themeit=THEME3)

addFile('Clear Old Thumbnails', 'oldThumbs', icon=ICONMAINT, themeit=THEME3)

addFile('Clear Crash Logs', 'clearcrash', icon=ICONMAINT, themeit=THEME3)

addFile('Purge Databases', 'purgedb', icon=ICONMAINT, themeit=THEME3)

addDir ('[B]Back up/Restore[/B]' , 'backup', icon=ICONMAINT, themeit=THEME1)

addDir ('[B]Advanced Settings Tool[/B]' , 'autoconfig', icon=ICONMAINT, themeit=THEME1)

addDir ('[B]Addon Tools[/B]', 'addon', icon=ICONMAINT, themeit=THEME1)

addDir ('[B]Misc Maintenance[/B]' , 'misc', icon=ICONMAINT, themeit=THEME1)

addDir ('[B]System Tweaks/Fixes[/B]', 'tweaks', icon=ICONMAINT, themeit=THEME1)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', themeit=THEME3)

addFile('!!!>>Fresh Start<<!!!', 'freshstart', icon=ICONMAINT, themeit=THEME6)

def backup():

addFile('Back Up Location: [COLOR %s]%s[/COLOR]' % (COLOR2, MYBUILDS),'settings', 'Maintenance', icon=ICONMAINT, themeit=THEME3)

if HIDESPACERS == 'No': addFile(wiz.sep('Backup'), '', themeit=THEME1)

addFile('[Back Up]: Build', 'backupbuild', icon=ICONMAINT, themeit=THEME3)

addFile('[Back Up]: GuiFix', 'backupgui', icon=ICONMAINT, themeit=THEME3)

addFile('[Back Up]: Theme', 'backuptheme', icon=ICONMAINT, themeit=THEME3)

addFile('[Back Up]: Addon Pack', 'backupaddonpack', icon=ICONMAINT, themeit=THEME3)

addFile('[Back Up]: Addon\_data', 'backupaddon', icon=ICONMAINT, themeit=THEME3)

if HIDESPACERS == 'No': addFile(wiz.sep('Restore'), '', themeit=THEME1)

addFile('[Restore]: Local Build', 'restorezip', icon=ICONMAINT, themeit=THEME3)

addFile('[Restore]: Local GuiFix', 'restoregui', icon=ICONMAINT, themeit=THEME3)

addFile('[Restore]: Local Addon\_data', 'restoreaddon', icon=ICONMAINT, themeit=THEME3)

addFile('[Restore]: External Build', 'restoreextzip', icon=ICONMAINT, themeit=THEME3)

addFile('[Restore]: External GuiFix', 'restoreextgui', icon=ICONMAINT, themeit=THEME3)

addFile('[Restore]: External Addon\_data', 'restoreextaddon', icon=ICONMAINT, themeit=THEME3)

if HIDESPACERS == 'No': addFile(wiz.sep('Delete All Backups'), '', themeit=THEME1)

addFile('Clean Up Back Up Folder', 'clearbackup', icon=ICONMAINT, themeit=THEME3)

def addon():

addFile('Remove Addons', 'removeaddons', icon=ICONMAINT, themeit=THEME3)

addDir ('Remove Addon Data', 'removeaddondata', icon=ICONMAINT, themeit=THEME3)

addDir ('Enable/Disable Addons', 'enableaddons', icon=ICONMAINT, themeit=THEME3)

addFile('Enable/Disable Adult Addons', 'toggleadult', icon=ICONMAINT, themeit=THEME3)

addFile('Force Update Addons', 'forceupdate', icon=ICONMAINT, themeit=THEME3)

addFile('Hide Passwords On Keyboard Entry', 'hidepassword', icon=ICONMAINT, themeit=THEME3)

addFile('Unhide Passwords On Keyboard Entry', 'unhidepassword', icon=ICONMAINT, themeit=THEME3)

def misc():

errors = int(errorChecking(count=True))

err = str(errors)

errorsfound = '[COLOR red]%s[/COLOR] Error(s) Found' % (err) if errors > 0 else 'None Found'

wizlogsize = ': [COLOR red]Not Found[/COLOR]' if not os.path.exists(WIZLOG) else ": [COLOR green]%s[/COLOR]" % wiz.convertSize(os.path.getsize(WIZLOG))

addFile('Kodi 17 Fix', 'kodi17fix', icon=ICONMAINT, themeit=THEME3)

addDir ('Speed Test', 'speedtest', icon=ICONMAINT, themeit=THEME3)

addFile('Enable Unknown Sources', 'unknownsources', icon=ICONMAINT, themeit=THEME3)

addFile('Reload Skin', 'forceskin', icon=ICONMAINT, themeit=THEME3)

addFile('Reload Profile', 'forceprofile', icon=ICONMAINT, themeit=THEME3)

addFile('Force Close Kodi', 'forceclose', icon=ICONMAINT, themeit=THEME3)

addFile('Upload Log File', 'uploadlog', icon=ICONMAINT, themeit=THEME3)

addFile('View Errors in Log: %s' % (errorsfound), 'viewerrorlog', icon=ICONMAINT, themeit=THEME3)

if errors > 0: addFile('View Last Error In Log', 'viewerrorlast', icon=ICONMAINT, themeit=THEME3)

addFile('View Log File', 'viewlog', icon=ICONMAINT, themeit=THEME3)

addFile('View Wizard Log File', 'viewwizlog', icon=ICONMAINT, themeit=THEME3)

addFile('Clear Wizard Log File%s' % wizlogsize,'clearwizlog', icon=ICONMAINT, themeit=THEME3)

def autoconfig():

if not ADVANCEDFILE == 'http://' and not ADVANCEDFILE == '':

addDir ('Advanced Settings', 'advancedsetting', icon=ICONMAINT, themeit=THEME3)

else:

if os.path.exists(ADVANCED):

addFile('View Current AdvancedSettings.xml', 'currentsettings', icon=ICONMAINT, themeit=THEME3)

addFile('Remove Current AdvancedSettings.xml', 'removeadvanced', icon=ICONMAINT, themeit=THEME3)

addFile('Quick Configure AdvancedSettings.xml', 'autoadvanced', icon=ICONMAINT, themeit=THEME3)

addFile('Full Configure AdvancedSettings.xml', 'autoadvanced1', icon=ICONMAINT, themeit=THEME3)

def tweaks():

on = '[B][COLOR green]ON[/COLOR][/B]'; off = '[B][COLOR red]OFF[/COLOR][/B]'

autoclean = 'true' if AUTOCLEANUP == 'true' else 'false'

cache = 'true' if AUTOCACHE == 'true' else 'false'

packages = 'true' if AUTOPACKAGES == 'true' else 'false'

thumbs = 'true' if AUTOTHUMBS == 'true' else 'false'

maint = 'true' if SHOWMAINT == 'true' else 'false'

includevid = 'true' if INCLUDEVIDEO == 'true' else 'false'

includeall = 'true' if INCLUDEALL == 'true' else 'false'

thirdparty = 'true' if THIRDPARTY == 'true' else 'false'

addDir ('System Information', 'systeminfo', icon=ICONMAINT, themeit=THEME1)

addFile('Scan Sources for broken links', 'checksources', icon=ICONMAINT, themeit=THEME3)

addFile('Scan For Broken Repositories', 'checkrepos', icon=ICONMAINT, themeit=THEME3)

addFile('Fix Addons Not Updating', 'fixaddonupdate', icon=ICONMAINT, themeit=THEME3)

addFile('Remove Non-Ascii filenames', 'asciicheck', icon=ICONMAINT, themeit=THEME3)

addFile('Convert Paths to special', 'convertpath', icon=ICONMAINT, themeit=THEME3)

addFile('Third Party Wizards: %s' % thirdparty.replace('true',on).replace('false',off) ,'togglesetting', 'enable3rd', fanart=FANART, icon=ICONMAINT, themeit=THEME1)

if thirdparty == 'true':

first = THIRD1NAME if not THIRD1NAME == '' else 'Not Set'

secon = THIRD2NAME if not THIRD2NAME == '' else 'Not Set'

third = THIRD3NAME if not THIRD3NAME == '' else 'Not Set'

addFile('Edit Third Party Wizard 1: [COLOR %s]%s[/COLOR]' % (COLOR2, first), 'editthird', '1', icon=ICONMAINT, themeit=THEME3)

addFile('Edit Third Party Wizard 2: [COLOR %s]%s[/COLOR]' % (COLOR2, secon), 'editthird', '2', icon=ICONMAINT, themeit=THEME3)

addFile('Edit Third Party Wizard 3: [COLOR %s]%s[/COLOR]' % (COLOR2, third), 'editthird', '3', icon=ICONMAINT, themeit=THEME3)

def net\_tools(view=None):

addDir ('Speed Tester' ,'speedtest', icon=ICONAPK, themeit=THEME1)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', themeit=THEME3)

addDir ('View IP Address & MAC Address', 'viewIP', icon=ICONMAINT, themeit=THEME1)

setView('files', 'viewType')

def viewIP():

infoLabel = ['Network.IPAddress',

'Network.MacAddress',]

data = []; x = 0

for info in infoLabel:

temp = wiz.getInfo(info)

y = 0

while temp == "Busy" and y < 10:

temp = wiz.getInfo(info); y += 1; wiz.log("%s sleep %s" % (info, str(y))); xbmc.sleep(200)

data.append(temp)

x += 1

config = wiz.getConfig()

ipfinal = '%(ip)s' % config['client'] #else 'Unknown'

provider = '%(isp)s' % config['client'] #else 'Unknown'

location = '%(country)s' % config['client'] #else 'Unknown'

addFile('[COLOR %s]IP Local:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, data[0]), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]IP Externo:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, ipfinal), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Provider:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, provider), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Location:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, location), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]MacAddress:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, data[1]), '', icon=ICONMAINT, themeit=THEME2)

def speedTest():

addFile('Run Speed Test', 'speed', icon=ICONMAINT, themeit=THEME3)

if os.path.exists(SPEEDTESTFOLD):

speedimg = glob.glob(os.path.join(SPEEDTESTFOLD, '\*.png'))

speedimg.sort(key=lambda f: os.path.getmtime(f), reverse=True)

if len(speedimg) > 0:

addFile('Clear Results', 'clearspeedtest', icon=ICONMAINT, themeit=THEME3)

addFile(wiz.sep('Previous Runs'), '', icon=ICONMAINT, themeit=THEME3)

for item in speedimg:

created = datetime.fromtimestamp(os.path.getmtime(item)).strftime('%m/%d/%Y %H:%M:%S')

img = item.replace(os.path.join(SPEEDTESTFOLD, ''), '')

addFile('[B]%s[/B]: [I]Ran %s[/I]' % (img, created), 'viewspeedtest', img, icon=ICONMAINT, themeit=THEME3)

def clearSpeedTest():

speedimg = glob.glob(os.path.join(SPEEDTESTFOLD, '\*.png'))

for file in speedimg:

wiz.removeFile(file)

def viewSpeedTest(img=None):

img = os.path.join(SPEEDTESTFOLD, img)

notify.speedTest(img)

def speed():

try:

found = speedtest.speedtest()

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

urlsplits = found[0].split('/')

dest = os.path.join(SPEEDTESTFOLD, urlsplits[-1])

urllib.urlretrieve(found[0], dest)

viewSpeedTest(urlsplits[-1])

except:

wiz.log("[Speed Test] Error Running Speed Test")

pass

def advancedWindow(url=None):

if not ADVANCEDFILE == 'http://':

if url == None:

TEMPADVANCEDFILE = wiz.textCache(uservar.ADVANCEDFILE)

if TEMPADVANCEDFILE == False: ADVANCEDWORKING = wiz.workingURL(uservar.ADVANCEDFILE)

else:

TEMPADVANCEDFILE = wiz.textCache(url)

if TEMPADVANCEDFILE == False: ADVANCEDWORKING = wiz.workingURL(url)

addFile('Configure AdvancedSettings.xml', 'autoadvanced1', icon=ICONMAINT, themeit=THEME3)

addFile('Quick Configure AdvancedSettings.xml', 'autoadvanced', icon=ICONMAINT, themeit=THEME3)

if os.path.exists(ADVANCED):

addFile('View Current AdvancedSettings.xml', 'currentsettings', icon=ICONMAINT, themeit=THEME3)

addFile('Remove Current AdvancedSettings.xml', 'removeadvanced', icon=ICONMAINT, themeit=THEME3)

if not TEMPADVANCEDFILE == False:

if HIDESPACERS == 'No': addFile(wiz.sep(), '', icon=ICONMAINT, themeit=THEME3)

link = TEMPADVANCEDFILE.replace('\n','').replace('\r','').replace('\t','')

match = re.compile('name="(.+?)".+?ection="(.+?)".+?rl="(.+?)".+?con="(.+?)".+?anart="(.+?)".+?escription="(.+?)"').findall(link)

if len(match) > 0:

for name, section, url, icon, fanart, description in match:

if section.lower() == "yes":

addDir ("[B]%s[/B]" % name, 'advancedsetting', url, description=description, icon=icon, fanart=fanart, themeit=THEME3)

else:

addFile(name, 'writeadvanced', name, url, description=description, icon=icon, fanart=fanart, themeit=THEME2)

else: wiz.log("[Advanced Settings] ERROR: Invalid Format.")

else: wiz.log("[Advanced Settings] URL not working: %s" % ADVANCEDWORKING)

else: wiz.log("[Advanced Settings] not Enabled")

def writeAdvanced(name, url):

ADVANCEDWORKING = wiz.workingURL(url)

if ADVANCEDWORKING == True:

if os.path.exists(ADVANCED): choice = DIALOG.yesno(ADDONTITLE, "[COLOR %s]Would you like to overwrite your current Advanced Settings with [COLOR %s]%s[/COLOR]?[/COLOR]" % (COLOR2, COLOR1, name), yeslabel="[B][COLOR green]Overwrite[/COLOR][/B]", nolabel="[B][COLOR red]Cancel[/COLOR][/B]")

else: choice = DIALOG.yesno(ADDONTITLE, "[COLOR %s]Would you like to download and install [COLOR %s]%s[/COLOR]?[/COLOR]" % (COLOR2, COLOR1, name), yeslabel="[B][COLOR green]Install[/COLOR][/B]", nolabel="[B][COLOR red]Cancel[/COLOR][/B]")

if choice == 1:

file = wiz.openURL(url)

f = open(ADVANCED, 'w');

f.write(file)

f.close()

DIALOG.ok(ADDONTITLE, '[COLOR %s]AdvancedSettings.xml file has been successfully written. Once you click okay it will force close kodi.[/COLOR]' % COLOR2)

wiz.killxbmc(True)

else: wiz.log("[Advanced Settings] install canceled"); wiz.LogNotify('[COLOR %s]%s[/COLOR]' % (COLOR1, ADDONTITLE), "[COLOR %s]Write Cancelled![/COLOR]" % COLOR2); return

else: wiz.log("[Advanced Settings] URL not working: %s" % ADVANCEDWORKING); wiz.LogNotify('[COLOR %s]%s[/COLOR]' % (COLOR1, ADDONTITLE), "[COLOR %s]URL Not Working[/COLOR]" % COLOR2)

def viewAdvanced():

if os.path.exists(ADVANCED):

f = open(ADVANCED)

a = f.read().replace('\t', ' ')

wiz.TextBox(ADDONTITLE, a)

f.close()

else: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), "[COLOR %s]AdvancedSettings.xml not found[/COLOR]")

def removeAdvanced():

if os.path.exists(ADVANCED):

wiz.removeFile(ADVANCED)

else: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), "[COLOR %s]AdvancedSettings.xml not found[/COLOR]")

def showAutoAdvanced():

notify.autoConfig2()

def showAutoAdvanced1():

notify.autoConfig()

def getIP():

config = wiz.getConfig()

ipfinal = '%(ip)s' % config['client']

provider = '%(isp)s' % config['client']

location = '%(country)s]' % config['client']

return ipfinal, provider, location

def systemInfo():

infoLabel = ['System.FriendlyName',

'System.BuildVersion',

'System.CpuUsage',

'System.ScreenMode',

'Network.IPAddress',

'Network.MacAddress',

'System.Uptime',

'System.TotalUptime',

'System.FreeSpace',

'System.UsedSpace',

'System.TotalSpace',

'System.Memory(free)',

'System.Memory(used)',

'System.Memory(total)']

data = []; x = 0

for info in infoLabel:

temp = wiz.getInfo(info)

y = 0

while temp == "Busy" and y < 10:

temp = wiz.getInfo(info); y += 1; wiz.log("%s sleep %s" % (info, str(y))); xbmc.sleep(200)

data.append(temp)

x += 1

storage\_free = data[8] if 'Una' in data[8] else wiz.convertSize(int(float(data[8][:-8]))\*1024\*1024)

storage\_used = data[9] if 'Una' in data[9] else wiz.convertSize(int(float(data[9][:-8]))\*1024\*1024)

storage\_total = data[10] if 'Una' in data[10] else wiz.convertSize(int(float(data[10][:-8]))\*1024\*1024)

ram\_free = wiz.convertSize(int(float(data[11][:-2]))\*1024\*1024)

ram\_used = wiz.convertSize(int(float(data[12][:-2]))\*1024\*1024)

ram\_total = wiz.convertSize(int(float(data[13][:-2]))\*1024\*1024)

exter\_ip, provider, location = getIP()

picture = []; music = []; video = []; programs = []; repos = []; scripts = []; skins = []

fold = glob.glob(os.path.join(ADDONS, '\*/'))

for folder in sorted(fold, key = lambda x: x):

foldername = os.path.split(folder[:-1])[1]

if foldername == 'packages': continue

xml = os.path.join(folder, 'addon.xml')

if os.path.exists(xml):

f = open(xml)

a = f.read()

prov = re.compile("<provides>(.+?)</provides>").findall(a)

if len(prov) == 0:

if foldername.startswith('skin'): skins.append(foldername)

elif foldername.startswith('repo'): repos.append(foldername)

else: scripts.append(foldername)

elif not (prov[0]).find('executable') == -1: programs.append(foldername)

elif not (prov[0]).find('video') == -1: video.append(foldername)

elif not (prov[0]).find('audio') == -1: music.append(foldername)

elif not (prov[0]).find('image') == -1: picture.append(foldername)

addFile('[B]Media Center Info:[/B]', '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Name:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, data[0]), '', icon=ICONMAINT, themeit=THEME3)

addFile('[COLOR %s]Version:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, data[1]), '', icon=ICONMAINT, themeit=THEME3)

addFile('[COLOR %s]Platform:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, wiz.platform().title()), '', icon=ICONMAINT, themeit=THEME3)

addFile('[COLOR %s]CPU Usage:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, data[2]), '', icon=ICONMAINT, themeit=THEME3)

addFile('[COLOR %s]Screen Mode:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, data[3]), '', icon=ICONMAINT, themeit=THEME3)

addFile('[B]Uptime:[/B]', '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Current Uptime:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, data[6]), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Total Uptime:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, data[7]), '', icon=ICONMAINT, themeit=THEME2)

addFile('[B]Local Storage:[/B]', '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Used Storage:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, storage\_free), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Free Storage:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, storage\_used), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Total Storage:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, storage\_total), '', icon=ICONMAINT, themeit=THEME2)

addFile('[B]Ram Usage:[/B]', '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Used Memory:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, ram\_free), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Free Memory:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, ram\_used), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Total Memory:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, ram\_total), '', icon=ICONMAINT, themeit=THEME2)

addFile('[B]Network:[/B]', '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Local IP:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, data[4]), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]IP Externo:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, exter\_ip), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Provider:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, provider), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Location:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, location), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]MacAddress:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, data[5]), '', icon=ICONMAINT, themeit=THEME2)

totalcount = len(picture) + len(music) + len(video) + len(programs) + len(scripts) + len(skins) + len(repos)

addFile('[B]Addons([COLOR %s]%s[/COLOR]):[/B]' % (COLOR1, totalcount), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Video Addons:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, str(len(video))), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Program Addons:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, str(len(programs))), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Music Addons:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, str(len(music))), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Picture Addons:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, str(len(picture))), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Repositories:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, str(len(repos))), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Skins:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, str(len(skins))), '', icon=ICONMAINT, themeit=THEME2)

addFile('[COLOR %s]Scripts/Modules:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, str(len(scripts))), '', icon=ICONMAINT, themeit=THEME2)

def saveMenu():

on = '[COLOR green]ON[/COLOR]'; off = '[COLOR red]OFF[/COLOR]'

trakt = 'true' if KEEPTRAKT == 'true' else 'false'

alluc = 'true' if KEEPALLUC == 'true' else 'false'

real = 'true' if KEEPREAL == 'true' else 'false'

login = 'true' if KEEPLOGIN == 'true' else 'false'

sources = 'true' if KEEPSOURCES == 'true' else 'false'

advanced = 'true' if KEEPADVANCED == 'true' else 'false'

profiles = 'true' if KEEPPROFILES == 'true' else 'false'

favourites = 'true' if KEEPFAVS == 'true' else 'false'

repos = 'true' if KEEPREPOS == 'true' else 'false'

super = 'true' if KEEPSUPER == 'true' else 'false'

whitelist = 'true' if KEEPWHITELIST == 'true' else 'false'

addFile('Keep My \'WhiteList\': %s' % whitelist.replace('true',on).replace('false',off) ,'togglesetting', 'keepwhitelist', icon=ICONSAVE, themeit=THEME1)

if whitelist == 'true':

addFile(' Edit My Whitelist', 'whitelist', 'edit', icon=ICONSAVE, themeit=THEME1)

addFile(' View My Whitelist', 'whitelist', 'view', icon=ICONSAVE, themeit=THEME1)

addFile(' Clear My Whitelist', 'whitelist', 'clear', icon=ICONSAVE, themeit=THEME1)

addFile(' Import My Whitelist', 'whitelist', 'import', icon=ICONSAVE, themeit=THEME1)

addFile(' Export My Whitelist', 'whitelist', 'export', icon=ICONSAVE, themeit=THEME1)

addDir ('Keep Favourites' ,'FavsMenu', icon=ICONREAL, themeit=THEME1)

addDir ('Keep Trakt Data', 'trakt', icon=ICONTRAKT, themeit=THEME1)

addDir ('Keep Real Debrid', 'realdebrid', icon=ICONREAL, themeit=THEME1)

addDir ('Keep Alluc Login', 'alluc', icon=ICONLOGIN, themeit=THEME1)

addDir ('Keep Login Info', 'login', icon=ICONLOGIN, themeit=THEME1)

addFile('Import Save Data', 'managedata', 'import', icon=ICONSAVE, themeit=THEME1)

addFile('Export Save Data', 'managedata', 'export', icon=ICONSAVE, themeit=THEME1)

addFile('- Click to toggle settings -', '', themeit=THEME3)

addFile('Save Trakt: %s' % trakt.replace('true',on).replace('false',off) ,'togglesetting', 'keeptrakt', icon=ICONTRAKT, themeit=THEME1)

addFile('Save Real Debrid: %s' % real.replace('true',on).replace('false',off) ,'togglesetting', 'keepdebrid', icon=ICONREAL, themeit=THEME1)

addFile('Save Alluc Login: %s' % alluc.replace('true',on).replace('false',off) ,'togglesetting', 'keepalluc', icon=ICONREAL, themeit=THEME1)

addFile('Save Login Info: %s' % login.replace('true',on).replace('false',off) ,'togglesetting', 'keeplogin', icon=ICONLOGIN, themeit=THEME1)

#addFile('Keep \'Sources.xml\': %s' % sources.replace('true',on).replace('false',off) ,'togglesetting', 'keepsources', icon=ICONSAVE, themeit=THEME1)

addFile('Keep \'Profiles.xml\': %s' % profiles.replace('true',on).replace('false',off) ,'togglesetting', 'keepprofiles', icon=ICONSAVE, themeit=THEME1)

addFile('Keep \'Advancedsettings.xml\': %s' % advanced.replace('true',on).replace('false',off) ,'togglesetting', 'keepadvanced', icon=ICONSAVE, themeit=THEME1)

addFile('Keep \'Favourites.xml\': %s' % favourites.replace('true',on).replace('false',off) ,'togglesetting', 'keepfavourites', icon=ICONSAVE, themeit=THEME1)

addFile('Keep Super Favourites: %s' % super.replace('true',on).replace('false',off) ,'togglesetting', 'keepsuper', icon=ICONSAVE, themeit=THEME1)

addFile('Keep Installed Repo\'s: %s' % repos.replace('true',on).replace('false',off) ,'togglesetting', 'keeprepos', icon=ICONSAVE, themeit=THEME1)

setView('files', 'viewType')

def FavsMenu():

on = '[COLOR green]ON[/COLOR]'; off = '[COLOR red]OFF[/COLOR]'

fav = '[COLOR green]ON[/COLOR]' if KEEPFAVS == 'true' else '[COLOR red]OFF[/COLOR]'

last = str(FAVSsave) if not FAVSsave == '' else 'Seus Favoritos nao foi salvo ainda.'

if HIDESPACERS == 'No': addFile(wiz.sep(), '', themeit=THEME3)

addFile('Salvar Favoritos: %s' % fav, 'togglesetting', 'keepfavourites', icon=ICONTRAKT, themeit=THEME3)

if KEEPFAVS == 'true': addFile('Last Save: %s' % str(last), '', icon=ICONTRAKT, themeit=THEME3)

if HIDESPACERS == 'No': addFile(wiz.sep('Backs up a copy'), '', themeit=THEME3)

addFile('Salvar Favoritos', 'savefav', icon=ICONTRAKT, themeit=THEME1)

addFile('Recuperar Favoritos', 'restorefav', icon=ICONTRAKT, themeit=THEME1)

addFile('Apagar o Backup de Favoritos', 'clearfav', icon=ICONTRAKT, themeit=THEME1)

setView('files', 'viewType')

def traktMenu():

trakt = '[COLOR green]ON[/COLOR]' if KEEPTRAKT == 'true' else '[COLOR red]OFF[/COLOR]'

last = str(TRAKTSAVE) if not TRAKTSAVE == '' else 'Trakt hasnt been saved yet.'

addFile('[I]Register FREE Account at http://trakt.tv[/I]', '', icon=ICONTRAKT, themeit=THEME3)

addFile('Save Trakt Data: %s' % trakt, 'togglesetting', 'keeptrakt', icon=ICONTRAKT, themeit=THEME3)

if KEEPTRAKT == 'true': addFile('Last Save: %s' % str(last), '', icon=ICONTRAKT, themeit=THEME3)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', icon=ICONTRAKT, themeit=THEME3)

for trakt in traktit.ORDER:

name = TRAKTID[trakt]['name']

path = TRAKTID[trakt]['path']

saved = TRAKTID[trakt]['saved']

file = TRAKTID[trakt]['file']

user = wiz.getS(saved)

auser = traktit.traktUser(trakt)

icon = TRAKTID[trakt]['icon'] if os.path.exists(path) else ICONTRAKT

fanart = TRAKTID[trakt]['fanart'] if os.path.exists(path) else FANART

menu = createMenu('saveaddon', 'Trakt', trakt)

menu2 = createMenu('save', 'Trakt', trakt)

menu.append((THEME2 % '%s Settings' % name, 'RunPlugin(plugin://%s/?mode=opensettings&name=%s&url=trakt)' % (ADDON\_ID, trakt)))

addFile('[+]-> %s' % name, '', icon=icon, fanart=fanart, themeit=THEME3)

if not os.path.exists(path): addFile('[COLOR red]Addon Data: Not Installed[/COLOR]', '', icon=icon, fanart=fanart, menu=menu)

elif not auser: addFile('[COLOR red]Addon Data: Not Registered[/COLOR]','authtrakt', trakt, icon=icon, fanart=fanart, menu=menu)

else: addFile('[COLOR green]Addon Data: %s[/COLOR]' % auser,'authtrakt', trakt, icon=icon, fanart=fanart, menu=menu)

if user == "":

if os.path.exists(file): addFile('[COLOR red]Saved Data: Save File Found(Import Data)[/COLOR]','importtrakt', trakt, icon=icon, fanart=fanart, menu=menu2)

else : addFile('[COLOR red]Saved Data: Not Saved[/COLOR]','savetrakt', trakt, icon=icon, fanart=fanart, menu=menu2)

else: addFile('[COLOR green]Saved Data: %s[/COLOR]' % user, '', icon=icon, fanart=fanart, menu=menu2)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', themeit=THEME3)

addFile('Save All Trakt Data', 'savetrakt', 'all', icon=ICONTRAKT, themeit=THEME3)

addFile('Recover All Saved Trakt Data', 'restoretrakt', 'all', icon=ICONTRAKT, themeit=THEME3)

addFile('Import Trakt Data', 'importtrakt', 'all', icon=ICONTRAKT, themeit=THEME3)

addFile('Clear All Saved Trakt Data', 'cleartrakt', 'all', icon=ICONTRAKT, themeit=THEME3)

addFile('Clear All Addon Data', 'addontrakt', 'all', icon=ICONTRAKT, themeit=THEME3)

setView('files', 'viewType')

def realMenu():

real = '[COLOR green]ON[/COLOR]' if KEEPREAL == 'true' else '[COLOR red]OFF[/COLOR]'

last = str(REALSAVE) if not REALSAVE == '' else 'Real Debrid hasnt been saved yet.'

addFile('[I]http://real-debrid.com is a PAID service.[/I]', '', icon=ICONREAL, themeit=THEME3)

addFile('Save Real Debrid Data: %s' % real, 'togglesetting', 'keepdebrid', icon=ICONREAL, themeit=THEME3)

if KEEPREAL == 'true': addFile('Last Save: %s' % str(last), '', icon=ICONREAL, themeit=THEME3)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', icon=ICONREAL, themeit=THEME3)

for debrid in debridit.ORDER:

name = DEBRIDID[debrid]['name']

path = DEBRIDID[debrid]['path']

saved = DEBRIDID[debrid]['saved']

file = DEBRIDID[debrid]['file']

user = wiz.getS(saved)

auser = debridit.debridUser(debrid)

icon = DEBRIDID[debrid]['icon'] if os.path.exists(path) else ICONREAL

fanart = DEBRIDID[debrid]['fanart'] if os.path.exists(path) else FANART

menu = createMenu('saveaddon', 'Debrid', debrid)

menu2 = createMenu('save', 'Debrid', debrid)

menu.append((THEME2 % '%s Settings' % name, 'RunPlugin(plugin://%s/?mode=opensettings&name=%s&url=debrid)' % (ADDON\_ID, debrid)))

addFile('[+]-> %s' % name, '', icon=icon, fanart=fanart, themeit=THEME3)

if not os.path.exists(path): addFile('[COLOR red]Addon Data: Not Installed[/COLOR]', '', icon=icon, fanart=fanart, menu=menu)

elif not auser: addFile('[COLOR red]Addon Data: Not Registered[/COLOR]','authdebrid', debrid, icon=icon, fanart=fanart, menu=menu)

else: addFile('[COLOR green]Addon Data: %s[/COLOR]' % auser,'authdebrid', debrid, icon=icon, fanart=fanart, menu=menu)

if user == "":

if os.path.exists(file): addFile('[COLOR red]Saved Data: Save File Found(Import Data)[/COLOR]','importdebrid', debrid, icon=icon, fanart=fanart, menu=menu2)

else : addFile('[COLOR red]Saved Data: Not Saved[/COLOR]','savedebrid', debrid, icon=icon, fanart=fanart, menu=menu2)

else: addFile('[COLOR green]Saved Data: %s[/COLOR]' % user, '', icon=icon, fanart=fanart, menu=menu2)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', themeit=THEME3)

addFile('Save All Real Debrid Data', 'savedebrid', 'all', icon=ICONREAL, themeit=THEME3)

addFile('Recover All Saved Real Debrid Data', 'restoredebrid', 'all', icon=ICONREAL, themeit=THEME3)

addFile('Import Real Debrid Data', 'importdebrid', 'all', icon=ICONREAL, themeit=THEME3)

addFile('Clear All Saved Real Debrid Data', 'cleardebrid', 'all', icon=ICONREAL, themeit=THEME3)

addFile('Clear All Addon Data', 'addondebrid', 'all', icon=ICONREAL, themeit=THEME3)

setView('files', 'viewType')

def allucMenu():

alluc = '[COLOR green]ON[/COLOR]' if KEEPALLUC == 'true' else '[COLOR red]OFF[/COLOR]'

last = str(ALLUCSAVE) if not ALLUCSAVE == '' else 'Alluc Login hasnt been saved yet.'

addFile('[I]http://accounts.alluc.com/ is a Free service.[/I]', '', icon=ICONLOGIN, themeit=THEME3)

addFile('Save Alluc Login Data: %s' % alluc, 'togglesetting', 'keepalluc', icon=ICONLOGIN, themeit=THEME3)

if KEEPALLUC == 'true': addFile('Last Save: %s' % str(last), '', icon=ICONLOGIN, themeit=THEME3)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', icon=ICONLOGIN, themeit=THEME3)

for alluc in allucit.ORDER:

name = ALLUCID[alluc]['name']

path = ALLUCID[alluc]['path']

saved = ALLUCID[alluc]['saved']

file = ALLUCID[alluc]['file']

user = wiz.getS(saved)

auser = allucit.allucUser(alluc)

icon = ALLUCID[alluc]['icon'] if os.path.exists(path) else ICONLOGIN

fanart = ALLUCID[alluc]['fanart'] if os.path.exists(path) else FANART

menu = createMenu('saveaddon', 'ALLUC', alluc)

menu2 = createMenu('save', 'Alluc', alluc)

menu.append((THEME2 % '%s Settings' % name, 'RunPlugin(plugin://%s/?mode=opensettings&name=%s&url=alluc)' % (ADDON\_ID, alluc)))

addFile('[+]-> %s' % name, '', icon=icon, fanart=fanart, themeit=THEME3)

if not os.path.exists(path): addFile('[COLOR red]Addon Data: Not Installed[/COLOR]', '', icon=icon, fanart=fanart, menu=menu)

elif not auser: addFile('[COLOR red]Addon Data: Not Registered[/COLOR]','authalluc', alluc, icon=icon, fanart=fanart, menu=menu)

else: addFile('[COLOR green]Addon Data: %s[/COLOR]' % auser,'authalluc', alluc, icon=icon, fanart=fanart, menu=menu)

if user == "":

if os.path.exists(file): addFile('[COLOR red]Saved Data: Save File Found(Import Data)[/COLOR]','importalluc', alluc, icon=icon, fanart=fanart, menu=menu2)

else : addFile('[COLOR red]Saved Data: Not Saved[/COLOR]','savealluc', alluc, icon=icon, fanart=fanart, menu=menu2)

else: addFile('[COLOR green]Saved Data: %s[/COLOR]' % user, '', icon=icon, fanart=fanart, menu=menu2)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', themeit=THEME3)

addFile('Save All Alluc Login Data', 'savealluc', 'all', icon=ICONREAL, themeit=THEME3)

addFile('Recover All Saved Alluc Login Data', 'restorealluc', 'all', icon=ICONREAL, themeit=THEME3)

addFile('Import Alluc Login Data', 'importalluc', 'all', icon=ICONREAL, themeit=THEME3)

addFile('Clear All Saved Alluc Login Data', 'clearalluc', 'all', icon=ICONREAL, themeit=THEME3)

addFile('Clear All Addon Data', 'addonalluc', 'all', icon=ICONREAL, themeit=THEME3)

setView('files', 'viewType')

def loginMenu():

login = '[COLOR green]ON[/COLOR]' if KEEPLOGIN == 'true' else '[COLOR red]OFF[/COLOR]'

last = str(LOGINSAVE) if not LOGINSAVE == '' else 'Login data hasnt been saved yet.'

addFile('[I]Several of these addons are PAID services.[/I]', '', icon=ICONLOGIN, themeit=THEME3)

addFile('Save Login Data: %s' % login, 'togglesetting', 'keeplogin', icon=ICONLOGIN, themeit=THEME3)

if KEEPLOGIN == 'true': addFile('Last Save: %s' % str(last), '', icon=ICONLOGIN, themeit=THEME3)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', icon=ICONLOGIN, themeit=THEME3)

for login in loginit.ORDER:

name = LOGINID[login]['name']

path = LOGINID[login]['path']

saved = LOGINID[login]['saved']

file = LOGINID[login]['file']

user = wiz.getS(saved)

auser = loginit.loginUser(login)

icon = LOGINID[login]['icon'] if os.path.exists(path) else ICONLOGIN

fanart = LOGINID[login]['fanart'] if os.path.exists(path) else FANART

menu = createMenu('saveaddon', 'Login', login)

menu2 = createMenu('save', 'Login', login)

menu.append((THEME2 % '%s Settings' % name, 'RunPlugin(plugin://%s/?mode=opensettings&name=%s&url=login)' % (ADDON\_ID, login)))

addFile('[+]-> %s' % name, '', icon=icon, fanart=fanart, themeit=THEME3)

if not os.path.exists(path): addFile('[COLOR red]Addon Data: Not Installed[/COLOR]', '', icon=icon, fanart=fanart, menu=menu)

elif not auser: addFile('[COLOR red]Addon Data: Not Registered[/COLOR]','authlogin', login, icon=icon, fanart=fanart, menu=menu)

else: addFile('[COLOR green]Addon Data: %s[/COLOR]' % auser,'authlogin', login, icon=icon, fanart=fanart, menu=menu)

if user == "":

if os.path.exists(file): addFile('[COLOR red]Saved Data: Save File Found(Import Data)[/COLOR]','importlogin', login, icon=icon, fanart=fanart, menu=menu2)

else : addFile('[COLOR red]Saved Data: Not Saved[/COLOR]','savelogin', login, icon=icon, fanart=fanart, menu=menu2)

else: addFile('[COLOR green]Saved Data: %s[/COLOR]' % user, '', icon=icon, fanart=fanart, menu=menu2)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', themeit=THEME3)

addFile('Save All Login Data', 'savelogin', 'all', icon=ICONLOGIN, themeit=THEME3)

addFile('Recover All Saved Login Data', 'restorelogin', 'all', icon=ICONLOGIN, themeit=THEME3)

addFile('Import Login Data', 'importlogin', 'all', icon=ICONLOGIN, themeit=THEME3)

addFile('Clear All Saved Login Data', 'clearlogin', 'all', icon=ICONLOGIN, themeit=THEME3)

addFile('Clear All Addon Data', 'addonlogin', 'all', icon=ICONLOGIN, themeit=THEME3)

setView('files', 'viewType')

def fixUpdate():

if KODIV < 17:

dbfile = os.path.join(DATABASE, wiz.latestDB('Addons'))

try:

os.remove(dbfile)

except Exception, e:

wiz.log("Unable to remove %s, Purging DB" % dbfile)

wiz.purgeDb(dbfile)

else:

if os.path.exists(os.path.join(USERDATA, 'autoexec.py')):

temp = os.path.join(USERDATA, 'autoexec\_temp.py')

if os.path.exists(temp): xbmcvfs.delete(temp)

xbmcvfs.rename(os.path.join(USERDATA, 'autoexec.py'), temp)

xbmcvfs.copy(os.path.join(PLUGIN, 'resources', 'libs', 'autoexec.py'), os.path.join(USERDATA, 'autoexec.py'))

dbfile = os.path.join(DATABASE, wiz.latestDB('Addons'))

try:

os.remove(dbfile)

except Exception, e:

wiz.log("Unable to remove %s, Purging DB" % dbfile)

wiz.purgeDb(dbfile)

wiz.killxbmc(True)

def removeAddonMenu():

fold = glob.glob(os.path.join(ADDONS, '\*/'))

addonnames = []; addonids = []

for folder in sorted(fold, key = lambda x: x):

foldername = os.path.split(folder[:-1])[1]

if foldername in EXCLUDES: continue

elif foldername in DEFAULTPLUGINS: continue

elif foldername == 'packages': continue

xml = os.path.join(folder, 'addon.xml')

if os.path.exists(xml):

f = open(xml)

a = f.read()

match = wiz.parseDOM(a, 'addon', ret='id')

addid = foldername if len(match) == 0 else match[0]

try:

add = xbmcaddon.Addon(id=addid)

addonnames.append(add.getAddonInfo('name'))

addonids.append(addid)

except:

pass

if len(addonnames) == 0:

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), "[COLOR %s]No Addons To Remove[/COLOR]" % COLOR2)

return

if KODIV > 16:

selected = DIALOG.multiselect("%s: Select the addons you wish to remove." % ADDONTITLE, addonnames)

else:

selected = []; choice = 0

tempaddonnames = ["-- Click here to Continue --"] + addonnames

while not choice == -1:

choice = DIALOG.select("%s: Select the addons you wish to remove." % ADDONTITLE, tempaddonnames)

if choice == -1: break

elif choice == 0: break

else:

choice2 = (choice-1)

if choice2 in selected:

selected.remove(choice2)

tempaddonnames[choice] = addonnames[choice2]

else:

selected.append(choice2)

tempaddonnames[choice] = "[B][COLOR %s]%s[/COLOR][/B]" % (COLOR1, addonnames[choice2])

if selected == None: return

if len(selected) > 0:

wiz.addonUpdates('set')

for addon in selected:

removeAddon(addonids[addon], addonnames[addon], True)

xbmc.sleep(500)

if INSTALLMETHOD == 1: todo = 1

elif INSTALLMETHOD == 2: todo = 0

else: todo = DIALOG.yesno(ADDONTITLE, "[COLOR %s]Would you like to [COLOR %s]Force close[/COLOR] kodi or [COLOR %s]Reload Profile[/COLOR]?[/COLOR]" % (COLOR2, COLOR1, COLOR1), yeslabel="[B][COLOR green]Reload Profile[/COLOR][/B]", nolabel="[B][COLOR red]Force Close[/COLOR][/B]")

if todo == 1: wiz.reloadFix('remove addon')

else: wiz.addonUpdates('reset'); wiz.killxbmc(True)

def removeAddonDataMenu():

if os.path.exists(ADDOND):

addFile('[COLOR red][B][REMOVE][/B][/COLOR] All Addon\_Data', 'removedata', 'all', themeit=THEME2)

addFile('[COLOR red][B][REMOVE][/B][/COLOR] All Addon\_Data for Uninstalled Addons', 'removedata', 'uninstalled', themeit=THEME2)

addFile('[COLOR red][B][REMOVE][/B][/COLOR] All Empty Folders in Addon\_Data', 'removedata', 'empty', themeit=THEME2)

addFile('[COLOR red][B][REMOVE][/B][/COLOR] %s Addon\_Data' % ADDONTITLE, 'resetaddon', themeit=THEME2)

if HIDESPACERS == 'No': addFile(wiz.sep(), '', themeit=THEME3)

fold = glob.glob(os.path.join(ADDOND, '\*/'))

for folder in sorted(fold, key = lambda x: x):

foldername = folder.replace(ADDOND, '').replace('\\', '').replace('/', '')

icon = os.path.join(folder.replace(ADDOND, ADDONS), 'icon.png')

fanart = os.path.join(folder.replace(ADDOND, ADDONS), 'fanart.png')

folderdisplay = foldername

replace = {'audio.':'[COLOR orange][AUDIO] [/COLOR]', 'metadata.':'[COLOR cyan][METADATA] [/COLOR]', 'module.':'[COLOR orange][MODULE] [/COLOR]', 'plugin.':'[COLOR blue][PLUGIN] [/COLOR]', 'program.':'[COLOR orange][PROGRAM] [/COLOR]', 'repository.':'[COLOR gold][REPO] [/COLOR]', 'script.':'[COLOR green][SCRIPT] [/COLOR]', 'service.':'[COLOR green][SERVICE] [/COLOR]', 'skin.':'[COLOR dodgerblue][SKIN] [/COLOR]', 'video.':'[COLOR orange][VIDEO] [/COLOR]', 'weather.':'[COLOR yellow][WEATHER] [/COLOR]'}

for rep in replace:

folderdisplay = folderdisplay.replace(rep, replace[rep])

if foldername in EXCLUDES: folderdisplay = '[COLOR green][B][PROTECTED][/B][/COLOR] %s' % folderdisplay

else: folderdisplay = '[COLOR red][B][REMOVE][/B][/COLOR] %s' % folderdisplay

addFile(' %s' % folderdisplay, 'removedata', foldername, icon=icon, fanart=fanart, themeit=THEME2)

else:

addFile('No Addon data folder found.', '', themeit=THEME3)

setView('files', 'viewType')

def enableAddons():

addFile("[I][B][COLOR red]!!Notice: Disabling Some Addons Can Cause Issues!![/COLOR][/B][/I]", '', icon=ICONMAINT)

fold = glob.glob(os.path.join(ADDONS, '\*/'))

x = 0

for folder in sorted(fold, key = lambda x: x):

foldername = os.path.split(folder[:-1])[1]

if foldername in EXCLUDES: continue

if foldername in DEFAULTPLUGINS: continue

addonxml = os.path.join(folder, 'addon.xml')

if os.path.exists(addonxml):

x += 1

fold = folder.replace(ADDONS, '')[1:-1]

f = open(addonxml)

a = f.read().replace('\n','').replace('\r','').replace('\t','')

match = wiz.parseDOM(a, 'addon', ret='id')

match2 = wiz.parseDOM(a, 'addon', ret='name')

try:

pluginid = match[0]

name = match2[0]

except:

continue

try:

add = xbmcaddon.Addon(id=pluginid)

state = "[COLOR green][Enabled][/COLOR]"

goto = "false"

except:

state = "[COLOR red][Disabled][/COLOR]"

goto = "true"

pass

icon = os.path.join(folder, 'icon.png') if os.path.exists(os.path.join(folder, 'icon.png')) else ICON

fanart = os.path.join(folder, 'fanart.jpg') if os.path.exists(os.path.join(folder, 'fanart.jpg')) else FANART

addFile("%s %s" % (state, name), 'toggleaddon', fold, goto, icon=icon, fanart=fanart)

f.close()

if x == 0:

addFile("No Addons Found to Enable or Disable.", '', icon=ICONMAINT)

setView('files', 'viewType')

def changeFeq():

feq = ['Every Startup', 'Every Day', 'Every Three Days', 'Every Weekly']

change = DIALOG.select("[COLOR %s]How often would you list to Auto Clean on Startup?[/COLOR]" % COLOR2, feq)

if not change == -1:

wiz.setS('autocleanfeq', str(change))

wiz.LogNotify('[COLOR %s]Auto Clean Up[/COLOR]' % COLOR1, '[COLOR %s]Fequency Now %s[/COLOR]' % (COLOR2, feq[change]))

def developer():

addFile('Skin Swap Popup', 'sswap', themeit=THEME1)

addFile('Create QR Code', 'createqr', themeit=THEME1)

addFile('Test Notifications', 'testnotify', themeit=THEME1)

addFile('Test Update', 'testupdate', themeit=THEME1)

addFile('Test First Run', 'testfirst', themeit=THEME1)

addFile('Test First Run Settings', 'testfirstrun', themeit=THEME1)

addFile('Test Auto ADV Settings', 'autoadvanced', themeit=THEME1)

setView('files', 'viewType')

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

###### Build Install ######

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

def buildWizard(name, type, theme=None, over=False):

if over == False:

testbuild = wiz.checkBuild(name, 'url')

if testbuild == False:

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), "[COLOR %s]Unabled to find build[/COLOR]" % COLOR2)

return

testworking = wiz.workingURL(testbuild)

if testworking == False:

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), "[COLOR %s]Build Zip Error: %s[/COLOR]" % (COLOR2, testworking))

return

if type == 'gui':

if name == BUILDNAME:

if over == True: yes = 1

else: yes = DIALOG.yesno(ADDONTITLE, '[COLOR %s]Would you like to apply the guifix for:' % COLOR2, '[COLOR %s]%s[/COLOR]?[/COLOR]' % (COLOR1, name), nolabel='[B][COLOR red]No, Cancel[/COLOR][/B]',yeslabel='[B][COLOR green]Apply Fix[/COLOR][/B]')

else:

yes = DIALOG.yesno("%s - [COLOR red]WARNING!![/COLOR]" % ADDONTITLE, "[COLOR %s][COLOR %s]%s[/COLOR] community build is not currently installed." % (COLOR2, COLOR1, name), "Would you like to apply the guiFix anyways?.[/COLOR]", nolabel='[B][COLOR red]No, Cancel[/COLOR][/B]',yeslabel='[B][COLOR green]Apply Fix[/COLOR][/B]')

if yes:

buildzip = wiz.checkBuild(name,'gui')

zipname = name.replace('\\', '').replace('/', '').replace(':', '').replace('\*', '').replace('?', '').replace('"', '').replace('<', '').replace('>', '').replace('|', '')

if not wiz.workingURL(buildzip) == True: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]GuiFix: Invalid Zip Url![/COLOR]' % COLOR2); return

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

DP.create(ADDONTITLE,'[COLOR %s][B]Downloading GuiFix:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, name),'', 'Please Wait')

lib=os.path.join(PACKAGES, '%s\_guisettings.zip' % zipname)

try: os.remove(lib)

except: pass

downloader.download(buildzip, lib, DP)

xbmc.sleep(500)

title = '[COLOR %s][B]Instalando:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, name)

DP.update(0, title,'', 'Por favor, aguarde...')

extract.all(lib,USERDATA,DP, title=title)

DP.close()

wiz.defaultSkin()

wiz.lookandFeelData('save')

if KODIV >= 17:

installed = grabAddons(lib)

wiz.addonDatabase(installed, 1, True)

if INSTALLMETHOD == 1: todo = 1

elif INSTALLMETHOD == 2: todo = 0

else: todo = DIALOG.yesno(ADDONTITLE, "[COLOR %s]The Gui fix has been installed. Would you like to Reload the profile or Force Close Kodi?[/COLOR]" % COLOR2, yeslabel="[B][COLOR red]Reload Profile[/COLOR][/B]", nolabel="[B][COLOR green]Force Close[/COLOR][/B]")

if todo == 1: wiz.reloadFix()

else: DIALOG.ok(ADDONTITLE, "[COLOR %s]Obrigado por escolher a Vikings Build, para salvar as alteracoes e finalizar a instalacao, voce agora precisa forcar o fechemanto do Kodi, por favor, pressione OK para encerrar o Kodi[/COLOR]" % COLOR2); wiz.killxbmc('true')

else:

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]GuiFix: Cancelled![/COLOR]' % COLOR2)

elif type == 'fresh':

freshStart(name)

elif type == 'normal':

if url == 'normal':

if KEEPTRAKT == 'true':

traktit.autoUpdate('all')

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

if KEEPREAL == 'true':

debridit.autoUpdate('all')

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

if KEEPALLUC == 'true':

allucit.autoUpdate('all')

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

if KEEPLOGIN == 'true':

loginit.autoUpdate('all')

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

temp\_kodiv = int(KODIV); buildv = int(float(wiz.checkBuild(name, 'kodi')))

if not temp\_kodiv == buildv:

if temp\_kodiv == 16 and buildv <= 15: warning = False

else: warning = True

else: warning = False

if warning == True:

yes\_pressed = DIALOG.yesno("%s - [COLOR red]WARNING!![/COLOR]" % ADDONTITLE, '[COLOR %s]There is a chance that the skin will not appear correctly' % COLOR2, 'When installing a %s build on a Kodi %s install' % (wiz.checkBuild(name, 'kodi'), KODIV), 'Would you still like to install: [COLOR %s]%s v%s[/COLOR]?[/COLOR]' % (COLOR1, name, wiz.checkBuild(name,'version')), nolabel='[B][COLOR red]No, Cancel[/COLOR][/B]',yeslabel='[B][COLOR green]Yes, Install[/COLOR][/B]')

else:

if not over == False: yes\_pressed = 1

else: yes\_pressed = DIALOG.yesno(ADDONTITLE, '[COLOR %s]Would you like to Download and Install:' % COLOR2, '[COLOR %s]%s v%s[/COLOR]?[/COLOR]' % (COLOR1, name, wiz.checkBuild(name,'version')), nolabel='[B][COLOR red]No, Cancel[/COLOR][/B]',yeslabel='[B][COLOR green]Yes, Install[/COLOR][/B]')

if yes\_pressed:

wiz.clearS('build')

#buildzips = []

#buildzip1 = wiz.checkBuild(name, 'url')

#buildzip2 = wiz.checkBuild(name, 'url2')

#buildzip3 = wiz.checkBuild(name, 'url3')

#if not buildzip1 == 'http://': buildzips.append("Server 1")

#if not buildzip2 == 'http://': buildzips.append("Server 2")

#if not buildzip3 == 'http://': buildzips.append("Server 3")

#if len(buildzips) > 1:

# select = DIALOG.select("Select the server to download the build from", buildzips)

# selected = buildzips[select]

# if select == -1 or selected == "Server 1":

# buildzip = buildzip1

# elif selected == "Server 2":

# buildzip = buildzip2

# elif selected == "Server 3":

# buildzip = buildzip3

#elif len(buildzips) == 1:

# if not buildzip1 == 'http://': buildzip = buildzip1

# elif not buildzip2 == 'http://': buildzip = buildzip2

# elif not buildzip3 == 'http://': buildzip = buildzip3

buildzip = wiz.checkBuild(name, 'url')

zipname = name.replace('\\', '').replace('/', '').replace(':', '').replace('\*', '').replace('?', '').replace('"', '').replace('<', '').replace('>', '').replace('|', '')

if not wiz.workingURL(buildzip) == True: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]Build Install: Invalid Zip Url![/COLOR]' % COLOR2); return

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

DP.create(ADDONTITLE,'[COLOR %s][B]Baixando:[/B][/COLOR] [COLOR %s]%s v%s[/COLOR]' % (COLOR2, COLOR1, name, wiz.checkBuild(name,'version')),'', 'Por favor, aguarde..')

lib=os.path.join(PACKAGES, '%s.zip' % zipname)

try: os.remove(lib)

except: pass

downloader.download(buildzip, lib, DP)

xbmc.sleep(500)

title = '[COLOR %s][B]Instalando:[/B][/COLOR] [COLOR %s]%s v%s[/COLOR]' % (COLOR2, COLOR1, name, wiz.checkBuild(name,'version'))

DP.update(0, title,'', 'por favor, aguarde...')

percent, errors, error = extract.all(lib,HOME,DP, title=title)

if int(float(percent)) > 0:

wiz.fixmetas()

wiz.lookandFeelData('save')

wiz.defaultSkin()

#wiz.addonUpdates('set')

wiz.setS('buildname', name)

wiz.setS('buildversion', wiz.checkBuild( name,'version'))

wiz.setS('buildtheme', '')

wiz.setS('latestversion', wiz.checkBuild( name,'version'))

wiz.setS('lastbuildcheck', str(NEXTCHECK))

wiz.setS('installed', 'true')

wiz.setS('extract', str(percent))

wiz.setS('errors', str(errors))

wiz.log('INSTALLED %s: [ERRORS:%s]' % (percent, errors))

try: os.remove(lib)

except: pass

if int(float(errors)) > 0:

yes=DIALOG.yesno(ADDONTITLE, '[COLOR %s][COLOR %s]%s v%s[/COLOR]' % (COLOR2, COLOR1, name, wiz.checkBuild(name,'version')), 'Completado: [COLOR %s]%s%s[/COLOR] [Erros:[COLOR %s]%s[/COLOR]]' % (COLOR1, percent, '%', COLOR1, errors), 'Would you like to view the errors?[/COLOR]', nolabel='[B][COLOR red]No Thanks[/COLOR][/B]', yeslabel='[B][COLOR green]View Errors[/COLOR][/B]')

if yes:

if isinstance(errors, unicode):

error = error.encode('utf-8')

wiz.TextBox(ADDONTITLE, error)

DP.close()

themefile = wiz.themeCount(name)

if not themefile == False:

buildWizard(name, 'theme')

if KODIV >= 17: wiz.addonDatabase(ADDON\_ID, 1)

DIALOG.ok(ADDONTITLE, "[COLOR %s]Para salvar as alteracoes e finalizar a instalacao, voce agora precisa forcar o fechamento do Kodi, por favor, pressione OK para encerrar o Kodi[/COLOR]" % COLOR2); wiz.killxbmc('true')

else:

if isinstance(errors, unicode):

error = error.encode('utf-8')

wiz.TextBox("%s: Error Installing Build" % ADDONTITLE, error)

else:

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]Build Install: Cancelled![/COLOR]' % COLOR2)

elif type == 'theme':

if theme == None:

themefile = wiz.checkBuild(name, 'theme')

themelist = []

if not themefile == 'http://' and wiz.workingURL(themefile) == True:

themelist = wiz.themeCount(name, False)

if len(themelist) > 0:

if DIALOG.yesno(ADDONTITLE, "[COLOR %s]The Build [COLOR %s]%s[/COLOR] comes with [COLOR %s]%s[/COLOR] different themes" % (COLOR2, COLOR1, name, COLOR1, len(themelist)), "Would you like to install one now?[/COLOR]", yeslabel="[B][COLOR green]Install Theme[/COLOR][/B]", nolabel="[B][COLOR red]Cancel Themes[/COLOR][/B]"):

wiz.log("Theme List: %s " % str(themelist))

ret = DIALOG.select(ADDONTITLE, themelist)

wiz.log("Theme install selected: %s" % ret)

if not ret == -1: theme = themelist[ret]; installtheme = True

else: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]Theme Install: Cancelled![/COLOR]' % COLOR2); return

else: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]Theme Install: Cancelled![/COLOR]' % COLOR2); return

else: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]Theme Install: None Found![/COLOR]' % COLOR2)

else: installtheme = DIALOG.yesno(ADDONTITLE, '[COLOR %s]Would you like to install the theme:' % COLOR2, '[COLOR %s]%s[/COLOR]' % (COLOR1, theme), 'for [COLOR %s]%s v%s[/COLOR]?[/COLOR]' % (COLOR1, name, wiz.checkBuild(name,'version')), yeslabel="[B][COLOR green]Install Theme[/COLOR][/B]", nolabel="[B][COLOR red]Cancel Themes[/COLOR][/B]")

if installtheme:

themezip = wiz.checkTheme(name, theme, 'url')

zipname = name.replace('\\', '').replace('/', '').replace(':', '').replace('\*', '').replace('?', '').replace('"', '').replace('<', '').replace('>', '').replace('|', '')

if not wiz.workingURL(themezip) == True: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]Theme Install: Invalid Zip Url![/COLOR]' % COLOR2); return False

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

DP.create(ADDONTITLE,'[COLOR %s][B]Baixando:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, theme),'', 'Por favor, aguarde..')

lib=os.path.join(PACKAGES, '%s.zip' % zipname)

try: os.remove(lib)

except: pass

downloader.download(themezip, lib, DP)

xbmc.sleep(500)

DP.update(0,"", "Installing %s " % name)

test = False

if url not in ["fresh", "normal"]:

test = testTheme(lib) if not wiz.currSkin() in ['skin.confluence', 'skin.estuary'] else False

test2 = testGui(lib) if not wiz.currSkin() in ['skin.confluence', 'skin.estuary'] else False

if test == True:

wiz.lookandFeelData('save')

swap = wiz.skinToDefault('Theme Install')

if swap == False: return False

xbmc.sleep(500)

title = '[COLOR %s][B]Installing Theme:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, theme)

DP.update(0, title,'', 'Please Wait')

percent, errors, error = extract.all(lib,HOME,DP, title=title)

wiz.setS('buildtheme', theme)

wiz.log('INSTALLED %s: [ERRORS:%s]' % (percent, errors))

DP.close()

if url not in ["fresh", "normal"]:

wiz.forceUpdate()

if KODIV >= 17:

installed = grabAddons(lib)

wiz.addonDatabase(installed, 1, True)

if test2 == True:

wiz.lookandFeelData('save')

wiz.defaultSkin()

gotoskin = wiz.getS('defaultskin')

wiz.swapSkins(gotoskin, "Theme Installer")

wiz.lookandFeelData('restore')

elif test == True:

switch = wiz.swapSkins(gotoskin, 'Theme Install')

if switch == False: return

wiz.lookandFeelData('restore')

else:

wiz.ebi("ReloadSkin()")

xbmc.sleep(1000)

wiz.ebi("Container.Refresh")

else:

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]Theme Install: Cancelled![/COLOR]' % COLOR2)

def thirdPartyInstall(name, url):

if not wiz.workingURL(url):

LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]Invalid URL for Build[/COLOR]' % COLOR2); return

type = DIALOG.yesno(ADDONTITLE, "[COLOR %s]Would you like to preform a [COLOR %s]Fresh Install[/COLOR] or [COLOR %s]Normal Install[/COLOR] for:[/COLOR]" % (COLOR2, COLOR1, COLOR1), "[COLOR %s]%s[/COLOR]" % (COLOR1, name), yeslabel="[B][COLOR green]Fresh Install[/COLOR][/B]", nolabel="[B][COLOR red]Normal Install[/COLOR][/B]")

if type == 1:

freshStart('third', True)

wiz.clearS('build')

zipname = name.replace('\\', '').replace('/', '').replace(':', '').replace('\*', '').replace('?', '').replace('"', '').replace('<', '').replace('>', '').replace('|', '')

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

DP.create(ADDONTITLE,'[COLOR %s][B]Baixando:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, name),'', 'Por favor, aguarde..')

lib=os.path.join(PACKAGES, '%s.zip' % zipname)

try: os.remove(lib)

except: pass

downloader.download(url, lib, DP)

xbmc.sleep(500)

title = '[COLOR %s][B]Instalando:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, name)

DP.update(0, title,'', 'Por favor, aguarde...')

percent, errors, error = extract.all(lib,HOME,DP, title=title)

if int(float(percent)) > 0:

wiz.fixmetas()

wiz.lookandFeelData('save')

wiz.defaultSkin()

#wiz.addonUpdates('set')

wiz.setS('installed', 'true')

wiz.setS('extract', str(percent))

wiz.setS('errors', str(errors))

wiz.log('INSTALLED %s: [ERRORS:%s]' % (percent, errors))

try: os.remove(lib)

except: pass

if int(float(errors)) > 0:

yes=DIALOG.yesno(ADDONTITLE, '[COLOR %s][COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, name), 'Completed: [COLOR %s]%s%s[/COLOR] [Errors:[COLOR %s]%s[/COLOR]]' % (COLOR1, percent, '%', COLOR1, errors), 'Would you like to view the errors?[/COLOR]', nolabel='[B][COLOR red]No Thanks[/COLOR][/B]',yeslabel='[B][COLOR green]View Errors[/COLOR][/B]')

if yes:

if isinstance(errors, unicode):

error = error.encode('utf-8')

wiz.TextBox(ADDONTITLE, error)

DP.close()

if KODIV >= 17: wiz.addonDatabase(ADDON\_ID, 1)

if INSTALLMETHOD == 1: todo = 1

elif INSTALLMETHOD == 2: todo = 0

else: todo = DIALOG.yesno(ADDONTITLE, "[COLOR %s]Would you like to [COLOR %s]Force close[/COLOR] kodi or [COLOR %s]Reload Profile[/COLOR]?[/COLOR]" % (COLOR2, COLOR1, COLOR1), yeslabel="[B][COLOR green]Reload Profile[/COLOR][/B]", nolabel="[B][COLOR red]Force Close[/COLOR][/B]")

if todo == 1: wiz.reloadFix()

else: wiz.killxbmc(True)

def testTheme(path):

zfile = zipfile.ZipFile(path)

for item in zfile.infolist():

wiz.log(str(item.filename))

if '/settings.xml' in item.filename:

return True

return False

def testGui(path):

zfile = zipfile.ZipFile(path)

for item in zfile.infolist():

if '/guisettings.xml' in item.filename:

return True

return False

def grabAddons(path):

zfile = zipfile.ZipFile(path)

addonlist = []

for item in zfile.infolist():

if str(item.filename).find('addon.xml') == -1: continue

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

if not info[-2] in addonlist:

addonlist.append(info[-2])

return addonlist

def apkInstaller(apk, url):

wiz.log(apk)

wiz.log(url)

if wiz.platform() == 'android' or DEVELOPER == 'true':

yes = DIALOG.yesno(ADDONTITLE, "[COLOR %s]Would you like to download and install:" % COLOR2, "[COLOR %s]%s[/COLOR]" % (COLOR1, apk), yeslabel="[B][COLOR green]Download[/COLOR][/B]", nolabel="[B][COLOR red]Cancel[/COLOR][/B]")

if not yes: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]ERROR: Install Cancelled[/COLOR]' % COLOR2); return

display = apk

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

if not wiz.workingURL(url) == True: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]APK Installer: Invalid Apk Url![/COLOR]' % COLOR2); return

DP.create(ADDONTITLE,'[COLOR %s][B]Baixando:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, display),'', 'Por favor, aguarde..')

lib=os.path.join(PACKAGES, "%s.apk" % apk.replace('\\', '').replace('/', '').replace(':', '').replace('\*', '').replace('?', '').replace('"', '').replace('<', '').replace('>', '').replace('|', ''))

try: os.remove(lib)

except: pass

downloader.download(url, lib, DP)

xbmc.sleep(100)

DP.close()

notify.apkInstaller(apk)

wiz.ebi('StartAndroidActivity("","android.intent.action.VIEW","application/vnd.android.package-archive","file:'+lib+'")')

else: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]ERROR: None Android Device[/COLOR]' % COLOR2)

def apkInstaller1(apk, url):

if wiz.platform() == 'android' or DEVELOPER == 'true':

yes = DIALOG.yesno(ADDONTITLE, "[COLOR %s]Would you like to download and install:" % COLOR2, "[COLOR %s]%s[/COLOR]" % (COLOR1, apk), yeslabel="[B][COLOR green]Download[/COLOR][/B]", nolabel="[B][COLOR red]Cancel[/COLOR][/B]")

if not yes: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]ERROR: Install Cancelled[/COLOR]' % COLOR2); return

display = apk

if yes:

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

if not wiz.workingURL(url) == True: wiz.LogNotify(ADDONTITLE, 'APK Installer: [COLOR red]Invalid Apk Url![/COLOR]'); return

DP.create(ADDONTITLE,'Baixando %s' % display,'', 'Por favor, aguarde..')

lib=os.path.join(PACKAGES, "%s.apk" % apk.replace('\\', '').replace('/', '').replace(':', '').replace('\*', '').replace('?', '').replace('"', '').replace('<', '').replace('>', '').replace('|', ''))

try: os.remove(lib)

except: pass

downloader.download(url, lib, DP)

xbmc.sleep(500)

DP.close()

DIALOG.ok(ADDONTITLE, "Launching the APK to be installed", "Follow the install process to complete.")

xbmc.executebuiltin('StartAndroidActivity("","android.intent.action.VIEW","application/vnd.android.package-archive","file:'+lib+'")')

else: wiz.LogNotify(ADDONTITLE, '[COLOR red]ERROR:[/COLOR] Install Cancelled')

else: wiz.LogNotify(ADDONTITLE, '[COLOR red]ERROR:[/COLOR] None Android Device')

def romInstaller(name, url):

myroms = xbmc.translatePath(BACKUPROMS)

if myroms == '':

if DIALOG.yesno(ADDONTITLE, "[COLOR %s]It seems that you do not have an extract location setup for Rom Packs" % COLOR2, "Would you like to set one?[/COLOR]", yeslabel="[COLOR green][B]Set Location[/B][/COLOR]", nolabel="[COLOR red][B]Cancel Download[/B][/COLOR]"):

wiz.openS()

myroms = wiz.getS('rompath')

if myroms == '': return

yes = DIALOG.yesno(ADDONTITLE, "[COLOR %s]Tem certeza que voce gostaria de baixar e extrair? [COLOR %s]%s[/COLOR] para:" % (COLOR2, COLOR1, name), "[COLOR %s]%s[/COLOR]" % (COLOR1, myroms), yeslabel="[B][COLOR green]Download[/COLOR][/B]", nolabel="[B][COLOR red]Cancelar[/COLOR][/B]")

if not yes: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]ERROR: Install Cancelled[/COLOR]' % COLOR2); return

display = name

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

if not wiz.workingURL(url) == True: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]APK Installer: Invalid Rom Url![/COLOR]' % COLOR2); return

DP.create(ADDONTITLE,'[COLOR %s][B]Baixando:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, display),'', 'Por favor, aguarde..')

lib=os.path.join(PACKAGES, "%s.zip" % name.replace('\\', '').replace('/', '').replace(':', '').replace('\*', '').replace('?', '').replace('"', '').replace('<', '').replace('>', '').replace('|', ''))

try: os.remove(lib)

except: pass

downloader.download(url, lib, DP)

xbmc.sleep(100)

percent, errors, error = extract.all(lib,myroms,DP, title=display)

try: os.remove(lib)

except: pass

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]Rom Pack Installed[/COLOR]' % COLOR2)

DP.close()

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

###### Misc Functions######

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

def createMenu(type, add, name):

if type == 'saveaddon':

menu\_items=[]

add2 = urllib.quote\_plus(add.lower().replace(' ', ''))

add3 = add.replace('Debrid', 'Real Debrid')

name2 = urllib.quote\_plus(name.lower().replace(' ', ''))

name = name.replace('url', 'URL Resolver')

menu\_items.append((THEME2 % name.title(), ' '))

menu\_items.append((THEME3 % 'Save %s Data' % add3, 'RunPlugin(plugin://%s/?mode=save%s&name=%s)' % (ADDON\_ID, add2, name2)))

menu\_items.append((THEME3 % 'Restore %s Data' % add3, 'RunPlugin(plugin://%s/?mode=restore%s&name=%s)' % (ADDON\_ID, add2, name2)))

menu\_items.append((THEME3 % 'Clear %s Data' % add3, 'RunPlugin(plugin://%s/?mode=clear%s&name=%s)' % (ADDON\_ID, add2, name2)))

elif type == 'save' :

menu\_items=[]

add2 = urllib.quote\_plus(add.lower().replace(' ', ''))

add3 = add.replace('Debrid', 'Real Debrid')

name2 = urllib.quote\_plus(name.lower().replace(' ', ''))

name = name.replace('url', 'URL Resolver')

menu\_items.append((THEME2 % name.title(), ' '))

menu\_items.append((THEME3 % 'Register %s' % add3, 'RunPlugin(plugin://%s/?mode=auth%s&name=%s)' % (ADDON\_ID, add2, name2)))

menu\_items.append((THEME3 % 'Save %s Data' % add3, 'RunPlugin(plugin://%s/?mode=save%s&name=%s)' % (ADDON\_ID, add2, name2)))

menu\_items.append((THEME3 % 'Restore %s Data' % add3, 'RunPlugin(plugin://%s/?mode=restore%s&name=%s)' % (ADDON\_ID, add2, name2)))

menu\_items.append((THEME3 % 'Import %s Data' % add3, 'RunPlugin(plugin://%s/?mode=import%s&name=%s)' % (ADDON\_ID, add2, name2)))

menu\_items.append((THEME3 % 'Clear Addon %s Data' % add3, 'RunPlugin(plugin://%s/?mode=addon%s&name=%s)' % (ADDON\_ID, add2, name2)))

elif type == 'install' :

menu\_items=[]

name2 = urllib.quote\_plus(name)

menu\_items.append((THEME2 % name, 'RunAddon(%s, ?mode=viewbuild&name=%s)' % (ADDON\_ID, name2)))

menu\_items.append((THEME3 % 'Fresh Install', 'RunPlugin(plugin://%s/?mode=install&name=%s&url=fresh)' % (ADDON\_ID, name2)))

menu\_items.append((THEME3 % 'Normal Install', 'RunPlugin(plugin://%s/?mode=install&name=%s&url=normal)' % (ADDON\_ID, name2)))

menu\_items.append((THEME3 % 'Apply guiFix', 'RunPlugin(plugin://%s/?mode=install&name=%s&url=gui)' % (ADDON\_ID, name2)))

menu\_items.append((THEME3 % 'Build Information', 'RunPlugin(plugin://%s/?mode=buildinfo&name=%s)' % (ADDON\_ID, name2)))

menu\_items.append((THEME2 % '%s Settings' % ADDONTITLE, 'RunPlugin(plugin://%s/?mode=settings)' % ADDON\_ID))

return menu\_items

def toggleCache(state):

cachelist = ['includevideo', 'includeall', 'includebob', 'includezen,' 'includephoenix', 'includespecto', 'includegenesis', 'includeexodus', 'includeonechan', 'includesalts', 'includesaltslite']

titlelist = ['Include Video Addons', 'Include All Addons', 'Include Bob', 'Include Zen', 'Include Phoenix', 'Include Specto', 'Include Genesis', 'Include Exodus', 'Include One Channel', 'Include Salts', 'Include Salts Lite HD']

if state in ['true', 'false']:

for item in cachelist:

wiz.setS(item, state)

else:

if not state in ['includevideo', 'includeall'] and wiz.getS('includeall') == 'true':

try:

item = titlelist[cachelist.index(state)]

DIALOG.ok(ADDONTITLE, "[COLOR %s]You will need to turn off [COLOR %s]Include All Addons[/COLOR] to disable[/COLOR] [COLOR %s]%s[/COLOR]" % (COLOR2, COLOR1, COLOR1, item))

except:

wiz.LogNotify("[COLOR %s]Toggle Cache[/COLOR]" % COLOR1, "[COLOR %s]Invalid id: %s[/COLOR]" % (COLOR2, state))

else:

new = 'true' if wiz.getS(state) == 'false' else 'false'

wiz.setS(state, new)

def viewLogFile():

mainlog = wiz.Grab\_Log(True)

oldlog = wiz.Grab\_Log(True, True)

which = 0; logtype = mainlog

if not oldlog == False and not mainlog == False:

which = DIALOG.select(ADDONTITLE, ["View %s" % mainlog.replace(LOG, ""), "View %s" % oldlog.replace(LOG, "")])

if which == -1: wiz.LogNotify('[COLOR %s]View Log[/COLOR]' % COLOR1, '[COLOR %s]View Log Cancelled![/COLOR]' % COLOR2); return

elif mainlog == False and oldlog == False:

wiz.LogNotify('[COLOR %s]View Log[/COLOR]' % COLOR1, '[COLOR %s]No Log File Found![/COLOR]' % COLOR2)

return

elif not mainlog == False: which = 0

elif not oldlog == False: which = 1

logtype = mainlog if which == 0 else oldlog

msg = wiz.Grab\_Log(False) if which == 0 else wiz.Grab\_Log(False, True)

wiz.TextBox("%s - %s" % (ADDONTITLE, logtype), msg)

def errorList(file):

errors = []

a=open(file).read()

b=a.replace('\n','[CR]').replace('\r','')

match = re.compile("-->Python callback/script returned the following error<--(.+?)-->End of Python script error report<--").findall(b)

for item in match:

errors.append(item)

return errors

def errorChecking(log=None, count=None, last=None):

errors = []; error1 = []; error2 = [];

if log == None:

curr = wiz.Grab\_Log(True, False)

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

if old == False and curr == False:

if count == None:

wiz.LogNotify('[COLOR %s]View Error Log[/COLOR]' % COLOR1, '[COLOR %s]No Log File Found![/COLOR]' % COLOR2)

return

else:

return 0

if not curr == False:

error1 = errorList(curr)

if not old == False:

error2 = errorList(old)

if len(error2) > 0:

for item in error2: errors = [item] + errors

if len(error1) > 0:

for item in error1: errors = [item] + errors

else:

error1 = errorList(log)

if len(error1) > 0:

for item in error1: errors = [item] + errors

if not count == None:

return len(errors)

elif len(errors) > 0:

if last == None:

i = 0; string = ''

for item in errors:

i += 1

string += "[B][COLOR red]ERROR NUMBER %s:[/B][/COLOR]%s\n" % (str(i), item.replace(HOME, '/').replace(' ', ''))

else:

string = "[B][COLOR red]Last Error in Log:[/B][/COLOR]%s\n" % (errors[0].replace(HOME, '/').replace(' ', ''))

wiz.TextBox("%s: Errors in Log" % ADDONTITLE, string)

else:

wiz.LogNotify('[COLOR %s]View Error Log[/COLOR]' % COLOR1, '[COLOR %s]No Errors Found![/COLOR]' % COLOR2)

def log\_tools():

errors = int(errorChecking(count=True))

err = str(errors)

errorsfound = '[COLOR red]%s[/COLOR] Found' % (err) if errors > 0 else 'None Found'

on = '[B][COLOR green]ON[/COLOR][/B]'; off = '[B][COLOR red]OFF[/COLOR][/B]'

if wiz.Grab\_Log(True) == False: kodilog = 0

else: kodilog = errorChecking(wiz.Grab\_Log(True), True)

if wiz.Grab\_Log(True, True) == False: kodioldlog = 0

else: kodioldlog = errorChecking(wiz.Grab\_Log(True,True), True)

errorsinlog = int(kodilog) + int(kodioldlog)

wizlogsize = ': [COLOR red]Not Found[/COLOR]' if not os.path.exists(WIZLOG) else ": [COLOR green]%s[/COLOR]" % wiz.convertSize(os.path.getsize(WIZLOG))

return errorsfound

ACTION\_PREVIOUS\_MENU = 10 ## ESC action

ACTION\_NAV\_BACK = 92 ## Backspace action

ACTION\_MOVE\_LEFT = 1 ## Left arrow key

ACTION\_MOVE\_RIGHT = 2 ## Right arrow key

ACTION\_MOVE\_UP = 3 ## Up arrow key

ACTION\_MOVE\_DOWN = 4 ## Down arrow key

ACTION\_MOUSE\_WHEEL\_UP = 104 ## Mouse wheel up

ACTION\_MOUSE\_WHEEL\_DOWN = 105 ## Mouse wheel down

ACTION\_MOVE\_MOUSE = 107 ## Down arrow key

ACTION\_SELECT\_ITEM = 7 ## Number Pad Enter

ACTION\_BACKSPACE = 110 ## ?

ACTION\_MOUSE\_LEFT\_CLICK = 100

ACTION\_MOUSE\_LONG\_CLICK = 108

def LogViewer(default=None):

class LogViewer(xbmcgui.WindowXMLDialog):

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

self.default = kwargs['default']

def onInit(self):

self.title = 101

self.msg = 102

self.scrollbar = 103

self.upload = 201

self.kodi = 202

self.kodiold = 203

self.wizard = 204

self.okbutton = 205

f = open(self.default, 'r')

self.logmsg = f.read()

f.close()

self.titlemsg = "%s: %s" % (ADDONTITLE, self.default.replace(LOG, '').replace(ADDONDATA, ''))

self.showdialog()

def showdialog(self):

self.getControl(self.title).setLabel(self.titlemsg)

self.getControl(self.msg).setText(wiz.highlightText(self.logmsg))

self.setFocusId(self.scrollbar)

def onClick(self, controlId):

if controlId == self.okbutton: self.close()

elif controlId == self.upload: self.close(); uploadLog.Main()

elif controlId == self.kodi:

newmsg = wiz.Grab\_Log(False)

filename = wiz.Grab\_Log(True)

if newmsg == False:

self.titlemsg = "%s: View Log Error" % ADDONTITLE

self.getControl(self.msg).setText("Log File Does Not Exists!")

else:

self.titlemsg = "%s: %s" % (ADDONTITLE, filename.replace(LOG, ''))

self.getControl(self.title).setLabel(self.titlemsg)

self.getControl(self.msg).setText(wiz.highlightText(newmsg))

self.setFocusId(self.scrollbar)

elif controlId == self.kodiold:

newmsg = wiz.Grab\_Log(False, True)

filename = wiz.Grab\_Log(True, True)

if newmsg == False:

self.titlemsg = "%s: View Log Error" % ADDONTITLE

self.getControl(self.msg).setText("Log File Does Not Exists!")

else:

self.titlemsg = "%s: %s" % (ADDONTITLE, filename.replace(LOG, ''))

self.getControl(self.title).setLabel(self.titlemsg)

self.getControl(self.msg).setText(wiz.highlightText(newmsg))

self.setFocusId(self.scrollbar)

elif controlId == self.wizard:

newmsg = wiz.Grab\_Log(False, False, True)

filename = wiz.Grab\_Log(True, False, True)

if newmsg == False:

self.titlemsg = "%s: View Log Error" % ADDONTITLE

self.getControl(self.msg).setText("Log File Does Not Exists!")

else:

self.titlemsg = "%s: %s" % (ADDONTITLE, filename.replace(ADDONDATA, ''))

self.getControl(self.title).setLabel(self.titlemsg)

self.getControl(self.msg).setText(wiz.highlightText(newmsg))

self.setFocusId(self.scrollbar)

def onAction(self, action):

if action == ACTION\_PREVIOUS\_MENU: self.close()

elif action == ACTION\_NAV\_BACK: self.close()

if default == None: default = wiz.Grab\_Log(True)

lv = LogViewer( "LogViewer.xml" , ADDON.getAddonInfo('path'), 'DefaultSkin', default=default)

lv.doModal()

del lv

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

# `7MM"""YMM MMP""MM""YMM .g8"""bgd #

# MM `7 P' MM `7 .dP' `M #

# MM d MM dM' ` #

# MM""MM MM MM #

# MM Y MM MM. `7MMF' #

# MM MM `Mb. MM #

# .JMML. .JMML. `"bmmmdPY #

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

def removeAddon(addon, name, over=False):

if not over == False:

yes = 1

else:

yes = DIALOG.yesno(ADDONTITLE, '[COLOR %s]Tem certeza que deseja exluir o ADDON:'% COLOR2, 'Name: [COLOR %s]%s[/COLOR]' % (COLOR1, name), 'ID: [COLOR %s]%s[/COLOR][/COLOR]' % (COLOR1, addon), yeslabel='[B][COLOR green]Remover Addon[/COLOR][/B]', nolabel='[B][COLOR red]Cancelar [/COLOR][/B]')

if yes == 1:

folder = os.path.join(ADDONS, addon)

wiz.log("Removing Addon %s" % addon)

wiz.cleanHouse(folder)

xbmc.sleep(200)

try: shutil.rmtree(folder)

except Exception ,e: wiz.log("Error removing %s" % addon, xbmc.LOGNOTICE)

removeAddonData(addon, name, over)

if over == False:

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

def removeAddonData(addon, name=None, over=False):

if addon == 'all':

if DIALOG.yesno(ADDONTITLE, '[COLOR %s]Would you like to remove [COLOR %s]ALL[/COLOR] addon data stored in you Userdata folder?[/COLOR]' % (COLOR2, COLOR1), yeslabel='[B][COLOR green]Remove Data[/COLOR][/B]', nolabel='[B][COLOR red]Don\'t Remove[/COLOR][/B]'):

wiz.cleanHouse(ADDOND)

else: wiz.LogNotify('[COLOR %s]Remove Addon Data[/COLOR]' % COLOR1, '[COLOR %s]Cancelled![/COLOR]' % COLOR2)

elif addon == 'uninstalled':

if DIALOG.yesno(ADDONTITLE, '[COLOR %s]Would you like to remove [COLOR %s]ALL[/COLOR] addon data stored in you Userdata folder for uninstalled addons?[/COLOR]' % (COLOR2, COLOR1), yeslabel='[B][COLOR green]Remove Data[/COLOR][/B]', nolabel='[B][COLOR red]Don\'t Remove[/COLOR][/B]'):

total = 0

for folder in glob.glob(os.path.join(ADDOND, '\*')):

foldername = folder.replace(ADDOND, '').replace('\\', '').replace('/', '')

if foldername in EXCLUDES: pass

elif os.path.exists(os.path.join(ADDONS, foldername)): pass

else: wiz.cleanHouse(folder); total += 1; wiz.log(folder); shutil.rmtree(folder)

wiz.LogNotify('[COLOR %s]Clean up Uninstalled[/COLOR]' % COLOR1, '[COLOR %s]%s Folders(s) Removed[/COLOR]' % (COLOR2, total))

else: wiz.LogNotify('[COLOR %s]Remove Addon Data[/COLOR]' % COLOR1, '[COLOR %s]Cancelled![/COLOR]' % COLOR2)

elif addon == 'empty':

if DIALOG.yesno(ADDONTITLE, '[COLOR %s]Would you like to remove [COLOR %s]ALL[/COLOR] empty addon data folders in you Userdata folder?[/COLOR]' % (COLOR2, COLOR1), yeslabel='[B][COLOR green]Remove Data[/COLOR][/B]', nolabel='[B][COLOR red]Don\'t Remove[/COLOR][/B]'):

total = wiz.emptyfolder(ADDOND)

wiz.LogNotify('[COLOR %s]Remove Empty Folders[/COLOR]' % COLOR1, '[COLOR %s]%s Folders(s) Removed[/COLOR]' % (COLOR2, total))

else: wiz.LogNotify('[COLOR %s]Remove Empty Folders[/COLOR]' % COLOR1, '[COLOR %s]Cancelled![/COLOR]' % COLOR2)

else:

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

if addon in EXCLUDES:

wiz.LogNotify("[COLOR %s]Protected Plugin[/COLOR]" % COLOR1, "[COLOR %s]Not allowed to remove Addon\_Data[/COLOR]" % COLOR2)

elif os.path.exists(addon\_data):

if DIALOG.yesno(ADDONTITLE, '[COLOR %s]Would you also like to remove the addon data for:[/COLOR]' % COLOR2, '[COLOR %s]%s[/COLOR]' % (COLOR1, addon), yeslabel='[B][COLOR green]Remove Data[/COLOR][/B]', nolabel='[B][COLOR red]Don\'t Remove[/COLOR][/B]'):

wiz.cleanHouse(addon\_data)

try:

shutil.rmtree(addon\_data)

except:

wiz.log("Error deleting: %s" % addon\_data)

else:

wiz.log('Addon data for %s was not removed' % addon)

wiz.refresh()

def restoreit(type):

if type == 'build':

x = freshStart('restore')

if x == False: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), "[COLOR %s]Local Restore Cancelled[/COLOR]" % COLOR2); return

if not wiz.currSkin() in ['skin.confluence', 'skin.estuary']:

wiz.skinToDefault('Restore Backup')

wiz.restoreLocal(type)

def restoreextit(type):

if type == 'build':

x = freshStart('restore')

if x == False: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), "[COLOR %s]External Restore Cancelled[/COLOR]" % COLOR2); return

wiz.restoreExternal(type)

def buildInfo(name):

if wiz.workingURL(BUILDFILE) == True:

if wiz.checkBuild(name, 'url'):

name, version, url, minor, gui, kodi, theme, icon, fanart, preview, adult, info, description = wiz.checkBuild(name, 'all')

adult = 'Yes' if adult.lower() == 'yes' else 'No'

extend = False

if not info == "http://":

try:

tname, extracted, zipsize, skin, created, programs, video, music, picture, repos, scripts = wiz.checkInfo(info)

extend = True

except:

extend = False

if extend == True:

msg = "[COLOR %s]Nome da Build:[/COLOR] [COLOR %s]%s[/COLOR][CR][CR]" % (COLOR2, COLOR1, name)

msg += "[COLOR %s]Versao de Build:[/COLOR] v[COLOR %s]%s[/COLOR][CR][CR]" % (COLOR2, COLOR1, version)

msg += "[COLOR %s]Ultima atualizacao:[/COLOR] [COLOR %s]%s[/COLOR][CR][CR]" % (COLOR2, COLOR1, created)

if not theme == "http://":

themecount = wiz.themeCount(name, False)

msg += "[COLOR %s]Build Theme(s):[/COLOR] [COLOR %s]%s[/COLOR][CR][CR]" % (COLOR2, COLOR1, ', '.join(themecount))

msg += "[COLOR %s]Versaod e Kodi Instalada:[/COLOR] [COLOR %s]%s[/COLOR][CR][CR]" % (COLOR2, COLOR1, kodi)

msg += "[COLOR %s]Extracted Size:[/COLOR] [COLOR %s]%s[/COLOR][CR][CR]" % (COLOR2, COLOR1, wiz.convertSize(int(float(extracted))))

msg += "[COLOR %s]Zip Size:[/COLOR] [COLOR %s]%s[/COLOR][CR][CR]" % (COLOR2, COLOR1, wiz.convertSize(int(float(zipsize))))

msg += "[COLOR %s]Skin Name:[/COLOR] [COLOR %s]%s[/COLOR][CR][CR]" % (COLOR2, COLOR1, skin)

msg += "[COLOR %s]Adult Content:[/COLOR] [COLOR %s]%s[/COLOR][CR][CR]" % (COLOR2, COLOR1, adult)

msg += "[COLOR %s]Description:[/COLOR] [COLOR %s]%s[/COLOR][CR][CR]" % (COLOR2, COLOR1, description)

msg += "[COLOR %s]Programs:[/COLOR] [COLOR %s]%s[/COLOR][CR][CR]" % (COLOR2, COLOR1, programs)

msg += "[COLOR %s]Video:[/COLOR] [COLOR %s]%s[/COLOR][CR][CR]" % (COLOR2, COLOR1, video)

msg += "[COLOR %s]Music:[/COLOR] [COLOR %s]%s[/COLOR][CR][CR]" % (COLOR2, COLOR1, music)

msg += "[COLOR %s]Pictures:[/COLOR] [COLOR %s]%s[/COLOR][CR][CR]" % (COLOR2, COLOR1, picture)

msg += "[COLOR %s]Repositories:[/COLOR] [COLOR %s]%s[/COLOR][CR][CR]" % (COLOR2, COLOR1, repos)

msg += "[COLOR %s]Scripts:[/COLOR] [COLOR %s]%s[/COLOR]" % (COLOR2, COLOR1, scripts)

else:

msg = "[COLOR %s]Build Name:[/COLOR] [COLOR %s]%s[/COLOR][CR]" % (COLOR2, COLOR1, name)

msg += "[COLOR %s]Build Version:[/COLOR] [COLOR %s]%s[/COLOR][CR]" % (COLOR2, COLOR1, version)

if not theme == "http://":

themecount = wiz.themeCount(name, False)

msg += "[COLOR %s]Build Theme(s):[/COLOR] [COLOR %s]%s[/COLOR][CR]" % (COLOR2, COLOR1, ', '.join(themecount))

msg += "[COLOR %s]Kodi Version:[/COLOR] [COLOR %s]%s[/COLOR][CR]" % (COLOR2, COLOR1, kodi)

msg += "[COLOR %s]Adult Content:[/COLOR] [COLOR %s]%s[/COLOR][CR]" % (COLOR2, COLOR1, adult)

msg += "[COLOR %s]Description:[/COLOR] [COLOR %s]%s[/COLOR][CR]" % (COLOR2, COLOR1, description)

wiz.TextBox(ADDONTITLE, msg)

else: wiz.log("Invalid Build Name!")

else: wiz.log("Build text file not working: %s" % WORKINGURL)

def viewpack():

WORKINGURL = wiz.workingURL(ADDONPACK)

if not WORKINGURL == True:

addFile('Url for txt file not valid', '', themeit=THEME3)

addFile('%s' % WORKINGURL, '', themeit=THEME3)

return

link = wiz.openURL(ADDONPACK).replace('\n','').replace('\r','').replace('\t','')

match = re.compile('name="(.+?)".+?rl="(.+?)".+?con="(.+?)".+?anart="(.+?)".+?escription="(.+?)"').findall(link)

for name, url, icon, fanart, description in match:

addFolder('',name,url,'addonpackwiz',icon,fanart,'')

def addonpackwiz():

yes = DIALOG.yesno(ADDONTITLE, '[COLOR %s]Would you like to install the:' % COLOR2, '[COLOR %s]%s[/COLOR]?[/COLOR]' % (COLOR1, name), nolabel='[B]No, Cancel[/B]',yeslabel='[B]Install[/B]')

if yes:

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

DP.create(ADDONTITLE,'[COLOR %s][B]Baixando:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, name),'', 'Por favor, aguarde..')

lib=os.path.join(PACKAGES, 'pack.zip')

try: os.remove(lib)

except: pass

downloader.download(url, lib, DP)

xbmc.sleep(500)

DP.update(0,"", "Installing %s " % name)

title = '[COLOR %s][B]Installing Addon Pack:[/B][/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, name)

DP.update(0, title,'', 'Please Wait')

percent, errors, error = extract.all(lib,HOME,DP, title=title)

wiz.log('INSTALLED %s: [ERRORS:%s]' % (percent, errors))

DP.close()

#DIALOG.ok(ADDONTITLE, "[COLOR %s]Addon Pack installed, you need to reset Kodi, Press OK to reset Kodi[/COLOR]" % COLOR2)

yes = DIALOG.yesno(ADDONTITLE, '[COLOR %s]Installed:' % COLOR2, '[COLOR %s]%s[/COLOR]?[/COLOR]' % (COLOR1, name),'[COLOR green]Successfully[/COLOR]', nolabel='[B]Reset[/B]',yeslabel='[B]Force Close[/B]')

if yes == 1:

wiz.killxbmc('true')

elif yes == 0:

wiz.RESET()

else:

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]Addon Pack: Cancelled![/COLOR]' % COLOR2)

def dependsList(plugin):

addonxml = os.path.join(ADDONS, plugin, 'addon.xml')

if os.path.exists(addonxml):

source = open(addonxml,mode='r'); link = source.read(); source.close();

match = wiz.parseDOM(link, 'import', ret='addon')

items = []

for depends in match:

if not 'xbmc.python' in depends:

items.append(depends)

return items

return []

def manageSaveData(do):

if do == 'import':

TEMP = os.path.join(ADDONDATA, 'temp')

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

source = DIALOG.browse(1, '[COLOR %s]Select the location of the SaveData.zip[/COLOR]' % COLOR2, 'files', '.zip', False, False, HOME)

if not source.endswith('.zip'):

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), "[COLOR %s]Import Data Error![/COLOR]" % (COLOR2))

return

tempfile = os.path.join(MYBUILDS, 'SaveData.zip')

goto = xbmcvfs.copy(source, tempfile)

wiz.log("%s" % str(goto))

extract.all(xbmc.translatePath(tempfile), TEMP)

trakt = os.path.join(TEMP, 'trakt')

login = os.path.join(TEMP, 'login')

debrid = os.path.join(TEMP, 'debrid')

x = 0

if os.path.exists(trakt):

x += 1

files = os.listdir(trakt)

if not os.path.exists(traktit.TRAKTFOLD): os.makedirs(traktit.TRAKTFOLD)

for item in files:

old = os.path.join(traktit.TRAKTFOLD, item)

temp = os.path.join(trakt, item)

if os.path.exists(old):

if not DIALOG.yesno(ADDONTITLE, "[COLOR %s]Would you like replace the current [COLOR %s]%s[/COLOR] file?" % (COLOR2, COLOR1, item), yeslabel="[B][COLOR green]Yes Replace[/COLOR][/B]", nolabel="[B][COLOR red]No Skip[/COLOR][/B]"): continue

else: os.remove(old)

shutil.copy(temp, old)

traktit.importlist('all')

traktit.traktIt('restore', 'all')

if os.path.exists(login):

x += 1

files = os.listdir(login)

if not os.path.exists(loginit.LOGINFOLD): os.makedirs(loginit.LOGINFOLD)

for item in files:

old = os.path.join(loginit.LOGINFOLD, item)

temp = os.path.join(login, item)

if os.path.exists(old):

if not DIALOG.yesno(ADDONTITLE, "[COLOR %s]Would you like replace the current [COLOR %s]%s[/COLOR] file?" % (COLOR2, COLOR1, item), yeslabel="[B][COLOR green]Yes Replace[/COLOR][/B]", nolabel="[B][COLOR red]No Skip[/COLOR][/B]"): continue

else: os.remove(old)

shutil.copy(temp, old)

loginit.importlist('all')

loginit.loginIt('restore', 'all')

if os.path.exists(debrid):

x += 1

files = os.listdir(debrid)

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

for item in files:

old = os.path.join(debridit.REALFOLD, item)

temp = os.path.join(debrid, item)

if os.path.exists(old):

if not DIALOG.yesno(ADDONTITLE, "[COLOR %s]Would you like replace the current [COLOR %s]%s[/COLOR] file?" % (COLOR2, COLOR1, item), yeslabel="[B][COLOR green]Yes Replace[/COLOR][/B]", nolabel="[B][COLOR red]No Skip[/COLOR][/B]"): continue

else: os.remove(old)

shutil.copy(temp, old)

debridit.importlist('all')

debridit.debridIt('restore', 'all')

wiz.cleanHouse(TEMP)

wiz.removeFolder(TEMP)

os.remove(tempfile)

if x == 0: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), "[COLOR %s]Save Data Import Failed[/COLOR]" % COLOR2)

else: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), "[COLOR %s]Save Data Import Complete[/COLOR]" % COLOR2)

elif do == 'export':

mybuilds = xbmc.translatePath(MYBUILDS)

dir = [traktit.TRAKTFOLD, debridit.REALFOLD, loginit.LOGINFOLD]

traktit.traktIt('update', 'all')

loginit.loginIt('update', 'all')

debridit.debridIt('update', 'all')

source = DIALOG.browse(3, '[COLOR %s]Select where you wish to export the savedata zip?[/COLOR]' % COLOR2, 'files', '', False, True, HOME)

source = xbmc.translatePath(source)

tempzip = os.path.join(mybuilds, 'SaveData.zip')

zipf = zipfile.ZipFile(tempzip, mode='w')

for fold in dir:

if os.path.exists(fold):

files = os.listdir(fold)

for file in files:

zipf.write(os.path.join(fold, file), os.path.join(fold, file).replace(ADDONDATA, ''), zipfile.ZIP\_DEFLATED)

zipf.close()

if source == mybuilds:

DIALOG.ok(ADDONTITLE, "[COLOR %s]Save data has been backed up to:[/COLOR]" % (COLOR2), "[COLOR %s]%s[/COLOR]" % (COLOR1, tempzip))

else:

try:

xbmcvfs.copy(tempzip, os.path.join(source, 'SaveData.zip'))

DIALOG.ok(ADDONTITLE, "[COLOR %s]Save data has been backed up to:[/COLOR]" % (COLOR2), "[COLOR %s]%s[/COLOR]" % (COLOR1, os.path.join(source, 'SaveData.zip')))

except:

DIALOG.ok(ADDONTITLE, "[COLOR %s]Save data has been backed up to:[/COLOR]" % (COLOR2), "[COLOR %s]%s[/COLOR]" % (COLOR1, tempzip))

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

###### Fresh Install ######

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

def freshStart(install=None, over=False):

if KEEPTRAKT == 'true':

traktit.autoUpdate('all')

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

if KEEPREAL == 'true':

debridit.autoUpdate('all')

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

if KEEPLOGIN == 'true':

loginit.autoUpdate('all')

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

if over == True: yes\_pressed = 1

elif install == 'restore': yes\_pressed=DIALOG.yesno(ADDONTITLE, "[COLOR %s]Click [B][COLOR springgreen] - Yes - [/COLOR][/B]" % COLOR2, "Para apagar sua Build Atual instalada, \r\nThen Install a Local or External Stored Build Back Up[/COLOR]", nolabel='[B][COLOR red]No, Cancel[/COLOR][/B]', yeslabel='[B][COLOR springgreen]Yes[/COLOR][/B]')

elif install: yes\_pressed=DIALOG.yesno(ADDONTITLE, "[COLOR %s]Clique em [B][COLOR springgreen] INSTALAR [/COLOR][/B]" % COLOR2, "Para iniciar o processo de instalacao da nova Build, sera necessario apagar a atual [COLOR %s]%s[/COLOR]!!" % (COLOR1, install), nolabel='[B][COLOR red]CANCELAR[/COLOR][/B]', yeslabel='[B][COLOR springgreen]instalar[/COLOR][/B]')

else: yes\_pressed=DIALOG.yesno(ADDONTITLE, "[COLOR %s]Do you wish to restore your" % COLOR2, "Configuration to default settings?[/COLOR]", nolabel='[B][COLOR red]No, Cancel[/COLOR][/B]', yeslabel='[B][COLOR springgreen]Yes[/COLOR][/B]')

if yes\_pressed:

if not wiz.currSkin() in ['skin.confluence', 'skin.estuary']:

skin = 'skin.confluence' if KODIV < 17 else 'skin.estuary'

#yes=DIALOG.yesno(ADDONTITLE, "[COLOR %s]The skin needs to be set back to [COLOR %s]%s[/COLOR]" % (COLOR2, COLOR1, skin[5:]), "Before doing a fresh install to clear all Texture files,", "Would you like us to do that for you?[/COLOR]", yeslabel="[B][COLOR springgreen]Switch Skins[/COLOR][/B]", nolabel="[B][COLOR red]I'll Do It[/COLOR][/B]";

#if yes:

skinSwitch.swapSkins(skin)

x = 0

xbmc.sleep(1000)

while not xbmc.getCondVisibility("Window.isVisible(yesnodialog)") and x < 150:

x += 1

xbmc.sleep(200)

wiz.ebi('SendAction(Select)')

if xbmc.getCondVisibility("Window.isVisible(yesnodialog)"):

wiz.ebi('SendClick(11)')

else: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]Fresh Install: Skin Swap Timed Out![/COLOR]' % COLOR2); return False

xbmc.sleep(1000)

if not wiz.currSkin() in ['skin.confluence', 'skin.estuary']:

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]Fresh Install: Skin Swap Failed![/COLOR]' % COLOR2)

return

wiz.addonUpdates('set')

xbmcPath=os.path.abspath(HOME)

DP.create(ADDONTITLE,"[COLOR %s]Calculating files and folders" % COLOR2,'', 'Please Wait![/COLOR]')

total\_files = sum([len(files) for r, d, files in os.walk(xbmcPath)]); del\_file = 0

DP.update(0, "[COLOR %s]Gathering Excludes list." % COLOR2)

EXCLUDES.append('My\_Builds')

EXCLUDES.append('archive\_cache')

if KEEPREPOS == 'true':

repos = glob.glob(os.path.join(ADDONS, 'repo\*/'))

for item in repos:

repofolder = os.path.split(item[:-1])[1]

if not repofolder == EXCLUDES:

EXCLUDES.append(repofolder)

if KEEPSUPER == 'true':

EXCLUDES.append('plugin.program.super.favourites')

if KEEPWHITELIST == 'true':

pvr = ''

whitelist = wiz.whiteList('read')

if len(whitelist) > 0:

for item in whitelist:

try: name, id, fold = item

except: pass

if fold.startswith('pvr'): pvr = id

depends = dependsList(fold)

for plug in depends:

if not plug in EXCLUDES:

EXCLUDES.append(plug)

depends2 = dependsList(plug)

for plug2 in depends2:

if not plug2 in EXCLUDES:

EXCLUDES.append(plug2)

if not fold in EXCLUDES:

EXCLUDES.append(fold)

if not pvr == '': wiz.setS('pvrclient', fold)

if wiz.getS('pvrclient') == '':

for item in EXCLUDES:

if item.startswith('pvr'):

wiz.setS('pvrclient', item)

DP.update(0, "[COLOR %s]Apagagando Arquivos e Pastas:" % COLOR2)

latestAddonDB = wiz.latestDB('Addons')

for root, dirs, files in os.walk(xbmcPath,topdown=True):

dirs[:] = [d for d in dirs if d not in EXCLUDES]

for name in files:

del\_file += 1

fold = root.replace('/','\\').split('\\')

x = len(fold)-1

if name == 'sources.xml' and fold[-1] == 'userdata' and KEEPSOURCES == 'true': wiz.log("Keep Sources: %s" % os.path.join(root, name), xbmc.LOGNOTICE)

elif name == 'favourites.xml' and fold[-1] == 'userdata' and KEEPFAVS == 'true': wiz.log("Keep Favourites: %s" % os.path.join(root, name), xbmc.LOGNOTICE)

elif name == 'profiles.xml' and fold[-1] == 'userdata' and KEEPPROFILES == 'true': wiz.log("Keep Profiles: %s" % os.path.join(root, name), xbmc.LOGNOTICE)

elif name == 'advancedsettings.xml' and fold[-1] == 'userdata' and KEEPADVANCED == 'true': wiz.log("Keep Advanced Settings: %s" % os.path.join(root, name), xbmc.LOGNOTICE)

elif name in LOGFILES: wiz.log("Keep Log File: %s" % name, xbmc.LOGNOTICE)

elif name.endswith('.db'):

try:

if name == latestAddonDB and KODIV >= 17: wiz.log("Ignoring %s on v%s" % (name, KODIV), xbmc.LOGNOTICE)

else: os.remove(os.path.join(root,name))

except Exception, e:

if not name.startswith('Textures13'):

wiz.log('Failed to delete, Purging DB', xbmc.LOGNOTICE)

wiz.log("-> %s" % (str(e)), xbmc.LOGNOTICE)

wiz.purgeDb(os.path.join(root,name))

else:

DP.update(int(wiz.percentage(del\_file, total\_files)), '', '[COLOR %s]File: [/COLOR][COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, name), '')

try: os.remove(os.path.join(root,name))

except Exception, e:

wiz.log("Error removing %s" % os.path.join(root, name), xbmc.LOGNOTICE)

wiz.log("-> / %s" % (str(e)), xbmc.LOGNOTICE)

if DP.iscanceled():

DP.close()

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

return False

for root, dirs, files in os.walk(xbmcPath,topdown=True):

dirs[:] = [d for d in dirs if d not in EXCLUDES]

for name in dirs:

DP.update(100, '', 'Cleaning Up Empty Folder: [COLOR %s]%s[/COLOR]' % (COLOR1, name), '')

if name not in ["Database","userdata","temp","addons","addon\_data"]:

shutil.rmtree(os.path.join(root,name),ignore\_errors=True, onerror=None)

if DP.iscanceled():

DP.close()

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

return False

DP.close()

wiz.clearS('build')

if over == True:

return True

elif install == 'restore':

return True

elif install:

buildWizard(install, 'normal', over=True)

else:

if INSTALLMETHOD == 1: todo = 1

elif INSTALLMETHOD == 2: todo = 0

else: todo = DIALOG.yesno(ADDONTITLE, "[COLOR %s]You Need To [COLOR %s]Force close[/COLOR] This App [COLOR %s]And[/COLOR] Then Restart It Again[/COLOR]" % (COLOR2, COLOR1, COLOR1), yeslabel="[B][COLOR white]>>>>>>[/COLOR][/B]", nolabel="[B][COLOR springgreen]Force Close[/COLOR][/B]")

if todo == 1: wiz.reloadFix('fresh')

else: wiz.addonUpdates('reset'); wiz.killxbmc(True)

else:

if not install == 'restore':

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]Fresh Install: Cancelled![/COLOR]' % COLOR2)

wiz.refresh()

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

###DELETE CACHE##############

####THANKS GUYS @ NaN #######

def clearCache():

if DIALOG.yesno(ADDONTITLE, '[COLOR %s]Gostaria de apagar o Cache?[/COLOR]' % COLOR2, nolabel='[B][COLOR red]Cancelar[/COLOR][/B]', yeslabel='[B][COLOR green]Limpar Cache[/COLOR][/B]'):

wiz.clearCache()

DC.setLabel('Tamanho: [B][COLOR lime]0.0 B[/B][/COLOR]')

def clearArchive():

if DIALOG.yesno(ADDONTITLE, '[COLOR %s]Voce Gostaria de Limpar o \'Archive\_Cache\' folder?[/COLOR]' % COLOR2, nolabel='[B][COLOR red]Cancelar[/COLOR][/B]', yeslabel='[B][COLOR green]Sim Limpar[/COLOR][/B]'):

wiz.clearArchive()

def clearPackages():

if DIALOG.yesno(ADDONTITLE, '[COLOR %s]Voce gostaria de apagar os arquivos temporarios?[/COLOR]' % COLOR2, nolabel='[B][COLOR red]Cancelar[/COLOR][/B]', yeslabel='[B][COLOR green]Apagar[/COLOR][/B]'):

wiz.clearPackages('total')

DPK.setLabel('Tamanho: [B][COLOR lime]0.0 B[/B][/COLOR]')

TPK.setLabel('Arquivos: [B][COLOR lime]0[/B][/COLOR]')

def totalClean():

if DIALOG.yesno(ADDONTITLE, '[COLOR %s]Voce gostaria de Limpar o Cache, Temporarios e Miniaturas?[/COLOR]' % COLOR2, nolabel='[B][COLOR red]Cancelar Processo[/COLOR][/B]',yeslabel='[B][COLOR green]Limpar Tudo[/COLOR][/B]'):

wiz.clearCache()

wiz.clearPackages('total')

clearThumb('total')

TC.setLabel('Tamanho: [B][COLOR lime]0.0 B[/B][/COLOR]')

DC.setLabel('Tamanho: [B][COLOR lime]0.0 B[/B][/COLOR]')

DPK.setLabel('Tamanho: [B][COLOR lime]0.0 B[/B][/COLOR]')

TPK.setLabel('Arquivos: [B][COLOR lime]0[/B][/COLOR]')

DTH.setLabel('Tamanho: [B][COLOR lime]0.0 B[/B][/COLOR]')

TTH.setLabel('Arquivos: [B][COLOR lime]0[/B][/COLOR]')

def clearThumb(type=None):

latest = wiz.latestDB('Textures')

if not type == None: choice = 1

else: choice = DIALOG.yesno(ADDONTITLE, '[COLOR %s]Deseja excluir o arquivo temporario %s e a pasta de miniaturas (temporaria)?' % (COLOR2, latest), "Isso nao ira danificar o KODI ou interferir no desempenho, sao e arquivos temporarios e sera criado na proxima entrada do Kodi, no momento eles apenas ocupam espaco de armazenamento![/COLOR]", nolabel='[B][COLOR red]Cancelar[/COLOR][/B]', yeslabel='[B][COLOR green]Apagar Miniaturas[/COLOR][/B]')

if choice == 1:

try: wiz.removeFile(os.join(DATABASE, latest))

except: wiz.log('Failed to delete, Purging DB.'); wiz.purgeDb(latest)

wiz.removeFolder(THUMBS)

DTH.setLabel('Size: [B][COLOR lime]0.0 B[/B][/COLOR]')

TTH.setLabel('Files: [B][COLOR lime]0[/B][/COLOR]')

else: wiz.log('Clear thumbnames cancelled')

wiz.redoThumbs()

def purgeDb():

DB = []; display = []

for dirpath, dirnames, files in os.walk(HOME):

for f in fnmatch.filter(files, '\*.db'):

if f != 'Thumbs.db':

found = os.path.join(dirpath, f)

DB.append(found)

dir = found.replace('\\', '/').split('/')

display.append('(%s) %s' % (dir[len(dir)-2], dir[len(dir)-1]))

if KODIV >= 16:

choice = DIALOG.multiselect("[COLOR %s]Select DB File to Purge[/COLOR]" % COLOR2, display)

if choice == None: wiz.LogNotify("[COLOR %s]Purge Database[/COLOR]" % COLOR1, "[COLOR %s]Cancelled[/COLOR]" % COLOR2)

elif len(choice) == 0: wiz.LogNotify("[COLOR %s]Purge Database[/COLOR]" % COLOR1, "[COLOR %s]Cancelled[/COLOR]" % COLOR2)

else:

for purge in choice: wiz.purgeDb(DB[purge])

else:

choice = DIALOG.select("[COLOR %s]Select DB File to Purge[/COLOR]" % COLOR2, display)

if choice == -1: wiz.LogNotify("[COLOR %s]Purge Database[/COLOR]" % COLOR1, "[COLOR %s]Cancelled[/COLOR]" % COLOR2)

else: wiz.purgeDb(DB[purge])

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

### DEVELOPER MENU #######

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

def testnotify():

url = wiz.workingURL(NOTIFICATION)

if url == True:

try:

id, msg = wiz.splitNotify(NOTIFICATION)

if id == False: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), "[COLOR %s]Notification: Not Formated Correctly[/COLOR]" % COLOR2); return

notify.notification(msg, True)

except Exception, e:

wiz.log("Error on Notifications Window: %s" % str(e), xbmc.LOGERROR)

else: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), "[COLOR %s]Invalid URL for Notification[/COLOR]" % COLOR2)

def testupdate():

if BUILDNAME == "":

notify.updateWindow()

else:

notify.updateWindow(BUILDNAME, BUILDVERSION, BUILDLATEST, wiz.checkBuild(BUILDNAME, 'icon'), wiz.checkBuild(BUILDNAME, 'fanart'))

def testfirst():

notify.firstRun()

def testfirstRun():

notify.firstRunSettings()

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

## Making the Directory####

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

def Add\_Directory\_Item(handle, url, listitem, isFolder):

xbmcplugin.addDirectoryItem(handle, url, listitem, isFolder)

def addDir2(name,url,mode,iconimage,fanart):

u=sys.argv[0]+"?url="+urllib.quote\_plus(url)+"&mode="+str(mode)+"&name="+urllib.quote\_plus(name)+"&iconimage="+urllib.quote\_plus(iconimage)

ok=True

liz=xbmcgui.ListItem(name, iconImage="DefaultFolder.png", thumbnailImage=iconimage)

liz.setInfo( type="Video", infoLabels={ "Title": name, "Plot": name } )

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

ok=xbmcplugin.addDirectoryItem(handle=int(sys.argv[1]),url=u,listitem=liz,isFolder=True)

return ok

def addFolder(type,name,url,mode,iconimage = '',FanArt = '',video = '',description = ''):

if type != 'folder2' and type != 'addon':

if len(iconimage) > 0:

iconimage = Images + iconimage

else:##F#T#G##

iconimage = 'DefaultFolder.png'

if type == 'addon':

if len(iconimage) > 0:

iconimage = iconimage

else:

iconimage = 'none'

if FanArt == '':

FanArt = FanArt

u=sys.argv[0]+"?url="+urllib.quote\_plus(url)+"&mode="+str(mode)+"&name="+urllib.quote\_plus(name)+"&FanArt="+urllib.quote\_plus(FanArt)+"&video="+urllib.quote\_plus(video)+"&description="+urllib.quote\_plus(description)

ok=True

liz=xbmcgui.ListItem(name, iconImage="DefaultFolder.png", thumbnailImage=iconimage)

liz.setInfo( type="Video", infoLabels={ "Title": name, "Plot": description } )

liz.setProperty( "FanArt\_Image", FanArt )

liz.setProperty( "Build.Video", video )

if (type=='folder') or (type=='folder2') or (type=='tutorial\_folder') or (type=='news\_folder'):

ok=Add\_Directory\_Item(handle=int(sys.argv[1]),url=u,listitem=liz,isFolder=True)

else:

ok=Add\_Directory\_Item(handle=int(sys.argv[1]),url=u,listitem=liz,isFolder=False)

return ok

def addDir(display, mode=None, name=None, url=None, menu=None, description=ADDONTITLE, overwrite=True, fanart=FANART, icon=ICON, themeit=None):

u = sys.argv[0]

if not mode == None: u += "?mode=%s" % urllib.quote\_plus(mode)

if not name == None: u += "&name="+urllib.quote\_plus(name)

if not url == None: u += "&url="+urllib.quote\_plus(url)

ok=True

if themeit: display = themeit % display

liz=xbmcgui.ListItem(display, iconImage="DefaultFolder.png", thumbnailImage=icon)

liz.setInfo( type="Video", infoLabels={ "Title": display, "Plot": description} )

liz.setProperty( "Fanart\_Image", fanart )

if not menu == None: liz.addContextMenuItems(menu, replaceItems=overwrite)

ok=xbmcplugin.addDirectoryItem(handle=int(sys.argv[1]),url=u,listitem=liz,isFolder=True)

return ok

def addFile(display, mode=None, name=None, url=None, menu=None, description=ADDONTITLE, overwrite=True, fanart=FANART, icon=ICON, themeit=None):

u = sys.argv[0]

if not mode == None: u += "?mode=%s" % urllib.quote\_plus(mode)

if not name == None: u += "&name="+urllib.quote\_plus(name)

if not url == None: u += "&url="+urllib.quote\_plus(url)

ok=True

if themeit: display = themeit % display

liz=xbmcgui.ListItem(display, iconImage="DefaultFolder.png", thumbnailImage=icon)

liz.setInfo( type="Video", infoLabels={ "Title": display, "Plot": description} )

liz.setProperty( "Fanart\_Image", fanart )

if not menu == None: liz.addContextMenuItems(menu, replaceItems=overwrite)

ok=xbmcplugin.addDirectoryItem(handle=int(sys.argv[1]),url=u,listitem=liz,isFolder=False)

return ok

def get\_params():

param=[]

paramstring=sys.argv[2]

if len(paramstring)>=2:

params=sys.argv[2]

cleanedparams=params.replace('?','')

if (params[len(params)-1]=='/'):

params=params[0:len(params)-2]

pairsofparams=cleanedparams.split('&')

param={}

for i in range(len(pairsofparams)):

splitparams={}

splitparams=pairsofparams[i].split('=')

if (len(splitparams))==2:

param[splitparams[0]]=splitparams[1]

return param

params=get\_params()

url=None

name=None

mode=None

try: mode=urllib.unquote\_plus(params["mode"])

except: pass

try: name=urllib.unquote\_plus(params["name"])

except: pass

try: url=urllib.unquote\_plus(params["url"])

except: pass

def setView(content, viewType):

if wiz.getS('auto-view')=='true':

views = wiz.getS(viewType)

if views == '50' and KODIV >= 17 and SKIN == 'skin.estuary': views = '55'

if views == '500' and KODIV >= 17 and SKIN == 'skin.estuary': views = '50'

wiz.ebi("Container.SetViewMode(%s)" % views)

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

################# GUI LAYOUT ######################

###### Dont be ASSHOLE and claim this #############

########### like you created it!! #################

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

wiz.FTGlog('<Must remain to get any support>')

FOCUS\_BUTTON\_COLOR = uservar.FOCUS\_BUTTON\_COLOR

DESCOLOR = uservar.DESCOLOR

DES\_T\_COLOR = uservar.DES\_T\_COLOR

MAIN\_BUTTONS\_TEXT = uservar.MAIN\_BUTTONS\_TEXT

OTHER\_BUTTONS\_TEXT = uservar.OTHER\_BUTTONS\_TEXT

LIST\_TEXT = uservar.LIST\_TEXT

HIGHLIGHT\_LIST = uservar.HIGHLIGHT\_LIST

net = net.Net()

window = pyxbmct.AddonDialogWindow('')

EXIT = os.path.join(ART , '%s.png' % uservar.EXIT\_BUTTON\_COLOR)

FBUTTON = os.path.join(ART , '%s.png' % FOCUS\_BUTTON\_COLOR)

LBUTTON = os.path.join(ART , '%s.png' % HIGHLIGHT\_LIST)

BUTTON = os.path.join(ART , 'button.png')

LISTBG = os.path.join(ART , 'listbg.png')

SPLASH = os.path.join(ART , 'splash.jpg')

SpeedBG = os.path.join(ART , 'speedtest.jpg')

MAINBG = os.path.join(ART , 'main.jpg')

NOTXT = os.path.join(ART , '%s.gif'% uservar.NO\_TXT\_FILE)

########

# TODO #

########

#

#

# Fix or add theme and gui install buttons

#

# Add cat to apks and addons

#

# Add Video Preview Popup

#

#

#

#

#

#

#

def playVideoB(url):

if 'watch?v=' in url:

a, b = url.split('?')

find = b.split('&')

for item in find:

if item.startswith('v='):

url = item[2:]

break

else: continue

elif 'embed' in url or 'youtu.be' in url:

a = url.split('/')

if len(a[-1]) > 5:

url = a[-1]

elif len(a[-2]) > 5:

url = a[-2]

wiz.log("YouTube URL: %s" % url)

notify.Preview(url)

def runspeedtest():

speed = speedtest.speedtest()

speedthumb.setImage(speed[0])

def HIDEALL():

try:

InstallButtonROM.setVisible(False)

except:pass

try:

InstallButtonEMU.setVisible(False)

except:pass

try:

no\_txt.setVisible(False)

except:pass

try:

splash.setVisible(False)

except:pass

try:

AddonButton.setVisible(False)

except:pass

try:

APKButton.setVisible(False)

except:pass

try:

ROMButton.setVisible(False)

except:pass

try:

EmuButton.setVisible(False)

except:pass

try:

InstallButton.setVisible(False)

except:pass

try:

FreshStartButton.setVisible(False)

except:pass

try:

listbg.setVisible(False)

except:pass

try:

no\_txt.setVisible(False)

except:pass

try:

splash.setVisible(False)

except:pass

try:

speedthumb.setVisible(False)

except:pass

try:

buildlist.setVisible(False)

except:pass

try:

PreviewButton.setVisible(False)

except:pass

try:

ThemeButton.setVisible(False)

except:pass

try:

buildinfobg.setVisible(False)

except:pass

try:

buildbg.setVisible(False)

except:pass

try:

buildthumb.setVisible(False)

except:pass

try:

buildtextbox.setVisible(False)

except:pass

try:

vertextbox .setVisible(False)

except:pass

try:

koditextbox.setVisible(False)

except:pass

try:

desctextbox.setVisible(False)

except:pass

try:

addthumb.setVisible(False)

except:pass

try:

InstallButtonA.setVisible(False)

except:pass

try:

addonlist.setVisible(False)

except:pass

try:

addthumb.setVisible(False)

except:pass

try:

desctextboxA.setVisible(False)

except:pass

try:

addtextbox.setVisible(False)

except:pass

try:

listbgA.setVisible(False)

except:pass

try:

buildbgA.setVisible(False)

except:pass

try:

apkthumb.setVisible(False)

except:pass

try:

apklist.setVisible(False)

except:pass

try:

apkthumb.setVisible(False)

except:pass

try:

apktextbox.setVisible(False)

except:pass

try:

desctextboxAPK.setVisible(False)

except:pass

try:

InstallButtonAPK.setVisible(False)

except:pass

try:

emulist.setVisible(False)

except:pass

try:

emuthumb.setVisible(False)

except:pass

try:

desctextboxEMU.setVisible(False)

except:pass

try:

emutextbox.setVisible(False)

except:pass

try:

InstallButtonEMU.setVisible(False)

except:pass

try:

romlist.setVisible(False)

except:pass

try:

romthumb.setVisible(False)

except:pass

try:

desctextboxROM.setVisible(False)

except:pass

try:

romtextbox.setVisible(False)

except:pass

try:

InstallButtonROM.setVisible(False)

except:pass

try:

maintbg.setVisible(False)

except:pass

try:

total\_clean\_button.setVisible(False)

except:pass

try:

total\_cache\_button.setVisible(False)

except:pass

try:

total\_packages\_button.setVisible(False)

except:pass

try:

total\_thumbnails\_button.setVisible(False)

except:pass

try:

TC.setVisible(False)

except:pass

try:

DC.setVisible(False)

except:pass

try:

DPK.setVisible(False)

except:pass

try:

DTH.setVisible(False)

except:pass

try:

TPK.setVisible(False)

except:pass

try:

TTH.setVisible(False)

except:pass

try:

sysinfobg.setVisible(False)

except:pass

try:

speedtest\_button.setVisible(False)

except:pass

try:

sysinfo\_title.setVisible(False)

except:pass

try:

version1.setVisible(False)

except:pass

try:

store.setVisible(False)

except:pass

try:

rom\_used.setVisible(False)

except:pass

try:

rom\_free.setVisible(False)

except:pass

try:

rom\_total.setVisible(False)

except:pass

try:

mem.setVisible(False)

except:pass

try:

ram\_used.setVisible(False)

except:pass

try:

ram\_free.setVisible(False)

except:pass

try:

ram\_total.setVisible(False)

except:pass

try:

kodi.setVisible(False)

except:pass

try:

total.setVisible(False)

except:pass

try:

video.setVisible(False)

except:pass

try:

program.setVisible(False)

except:pass

try:

music.setVisible(False)

except:pass

try:

picture.setVisible(False)

except:pass

try:

repos.setVisible(False)

except:pass

try:

skins.setVisible(False)

except:pass

try:

scripts.setVisible(False)

except:pass

try:

netinfobg.setVisible(False)

except:pass

try:

netinfo\_title.setVisible(False)

except:pass

try:

un\_hide\_net.setVisible(False)

except:pass

try:

settings\_button1.setVisible(False)

except:pass

try:

trigger\_title.setVisible(False)

except:pass

try:

MAC.setVisible(False)

except:pass

try:

INTER\_IP.setVisible(False)

except:pass

try:

IP.setVisible(False)

except:pass

try:

ISP.setVisible(False)

except:pass

try:

CITY.setVisible(False)

except:pass

try:

STATE.setVisible(False)

except:pass

try:

COUNTRY.setVisible(False)

except:pass

try:

bakresbg.setVisible(False)

except:pass

try:

favs.setVisible(False)

except:pass

try:

backuploc.setVisible(False)

except:pass

try:

Backup.setVisible(False)

except:pass

try:

backup\_build\_button.setVisible(False)

except:pass

try:

backup\_gui\_button.setVisible(False)

except:pass

try:

backup\_addondata\_button.setVisible(False)

except:pass

try:

restore\_build\_button.setVisible(False)

except:pass

try:

restore\_gui\_button.setVisible(False)

except:pass

try:

restore\_addondata\_button.setVisible(False)

except:pass

try:

clear\_backup\_button.setVisible(False)

except:pass

try:

savefav\_button.setVisible(False)

except:pass

try:

restorefav\_button.setVisible(False)

except:pass

try:

clearfav\_button.setVisible(False)

except:pass

try:

backupaddonpack\_button.setVisible(False)

except:pass

try:

restore\_title.setVisible(False)

except:pass

try:

delete\_title.setVisible(False)

except:pass

try:

set\_title.setVisible(False)

except:pass

try:

settings\_button.setVisible(False)

except:pass

try:

view\_error\_button.setVisible(False)

except:pass

try:

full\_log\_button.setVisible(False)

except:pass

try:

upload\_log\_button.setVisible(False)

except:pass

try:

removeaddons\_button.setVisible(False)

except:pass

try:

removeaddondata\_all\_button.setVisible(False)

except:pass

try:

removeaddondata\_u\_button.setVisible(False)

except:pass

try:

removeaddondata\_e\_button.setVisible(False)

except:pass

try:

checksources\_button.setVisible(False)

except:pass

try:

checkrepos\_button.setVisible(False)

except:pass

try:

forceupdate\_button.setVisible(False)

except:pass

try:

fixaddonupdate\_button.setVisible(False)

except:pass

try:

Addon.setVisible(False)

except:pass

try:

scan.setVisible(False)

except:pass

try:

fix.setVisible(False)

except:pass

try:

delet.setVisible(False)

except:pass

try:

delet1.setVisible(False)

except:pass

try:

Log\_title.setVisible(False)

except:pass

try:

toolsbg.setVisible(False)

except:pass

try:

Log\_errors.setVisible(False)

except:pass

try:

WhiteList.setVisible(False)

except:pass

try:

whitelist\_edit\_button.setVisible(False)

except:pass

try:

whitelist\_view\_button.setVisible(False)

except:pass

try:

whitelist\_clear\_button.setVisible(False)

except:pass

try:

whitelist\_import\_button.setVisible(False)

except:pass

try:

whitelist\_export\_button.setVisible(False)

except:pass

try:

Advan.setVisible(False)

except:pass

try:

autoadvanced\_buttonQ.setVisible(False)

except:pass

try:

autoadvanced\_button.setVisible(False)

except:pass

try:

currentsettings\_button.setVisible(False)

except:pass

try:

removeadvanced\_button.setVisible(False)

except:pass

try:

buildname.setVisible(False)

except:pass

try:

buildversion.setVisible(False)

except:pass

try:

buildtheme.setVisible(False)

except:pass

try:

skinname.setVisible(False)

except:pass

try:

errorinstall.setVisible(False)

except:pass

try:

lastupdatchk.setVisible(False)

except:pass

def list\_update():

global Bname

global url

global name

global plugin

try:

if window.getFocus() == buildlist:

pos=buildlist.getSelectedPosition()

link = net.http\_GET(BUILDFILE).content.replace('\n','').replace('\r','')

url = re.compile('url="(.+?)"').findall(link)[pos]

name = re.compile('name="(.+?)"').findall(link)[pos]

buildpic = re.compile('icon="(.+?)"').findall(link)[pos]

Bversion = re.compile('version="(.+?)"').findall(link)[pos]

kodivers = re.compile('kodi="(.+?)"').findall(link)[pos]

description = re.compile('description="(.+?)"').findall(link)[pos]

buildtextbox.setLabel('[COLOR %s]Build selecionada para instalar: [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR, name))

vertextbox.setLabel('[COLOR %s]Versao da Build: [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR, Bversion))

koditextbox.setLabel('[COLOR %s]Versaod e Kodi Instalado: [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR,kodivers))

desctextbox.setText('[COLOR %s]Descricao da Build: [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR,description))

buildthumb.setImage(buildpic)

Cname = buildlist.getListItem(buildlist.getSelectedPosition()).getLabel()

Bname = wiz.stripcolortags(Cname)

except:pass

try:

if window.getFocus() == addonlist:

pos=addonlist.getSelectedPosition()

link = net.http\_GET(ADDONFILE).content.replace('\n','').replace('\r','')

addpic = re.compile('icon="(.+?)"').findall(link)[pos]

url = re.compile('url="(.+?)"').findall(link)[pos]

name = re.compile('name="(.+?)"').findall(link)[pos]

description = re.compile('description="(.+?)"').findall(link)[pos]

plugin = re.compile('plugin="(.+?)"').findall(link)[pos]

addtextbox.setLabel('[COLOR %s]Addon Selecionado:[COLOR %s] %s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR,name))

desctextboxA.setText('[COLOR %s]Informacoes do ADDON:[COLOR %s] %s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR,description))

addthumb.setImage(addpic)

Cname = addonlist.getListItem(addonlist.getSelectedPosition()).getLabel()

name = wiz.stripcolortags(Cname)

except:pass

try:

if window.getFocus() == apklist:

pos=apklist.getSelectedPosition(name,url)

link = net.http\_GET(APKFILE).content.replace('\n','').replace('\r','')

apkpic = re.compile('icon="(.+?)"').findall(link)[pos]

url = re.compile('url="(.+?)"').findall(link)[pos]

name = re.compile('name="(.+?)"').findall(link)[pos]

description = re.compile('description="(.+?)"').findall(link)[pos]

apktextbox.setLabel('[COLOR %s]APK Selected:[COLOR %s] %s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR,name))

desctextboxAPK.setText('[COLOR %s]APK Description:[COLOR %s] %s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR,description))

apkthumb.setImage(apkpic)

Cname = apklist.getListItem(apklist.getSelectedPosition()).getLabel()

name = wiz.stripcolortags(Cname)

except:pass

try:

if window.getFocus() == emulist:

pos=emulist.getSelectedPosition(name,url)

link = net.http\_GET(EMUAPKS).content.replace('\n','').replace('\r','')

emupic = re.compile('icon="(.+?)"').findall(link)[pos]

url = re.compile('url="(.+?)"').findall(link)[pos]

name = re.compile('name="(.+?)"').findall(link)[pos]

description = re.compile('description="(.+?)"').findall(link)[pos]

emutextbox.setLabel('[COLOR %s]EMU Selected:[COLOR %s] %s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR,name))

desctextboxEMU.setText('[COLOR %s]EMU Description:[COLOR %s] %s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR,description))

emuthumb.setImage(emupic)

except:pass

try:

if window.getFocus() == romlist:

pos=romlist.getSelectedPosition(name,url)

link = net.http\_GET(ROMPACK).content.replace('\n','').replace('\r','')

emupic = re.compile('icon="(.+?)"').findall(link)[pos]

url = re.compile('url="(.+?)"').findall(link)[pos]

name = re.compile('name="(.+?)"').findall(link)[pos]

description = re.compile('description="(.+?)"').findall(link)[pos]

romtextbox.setLabel('[COLOR %s]ROM PACK Selected:[COLOR %s] %s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR,name))

desctextboxROM.setText('[COLOR %s]ROM PACK Description:[COLOR %s] %s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR,description))

romthumb.setImage(emupic)

except:pass

def BuildList():

global InstallButton

global FreshStartButton

global buildlist

global buildthumb

global buildtextbox

global vertextbox

global koditextbox

global desctextbox

global no\_txt

global buildname

global buildversion

global buildtheme

global skinname

global errorinstall

global lastupdatchk

global Bname

global PreviewButton

global ThemeButton

HIDEALL()

if not BUILDFILE == 'http://' and not BUILDFILE == '':

listbg.setVisible(True)

buildbg.setVisible(True)

buildinfobg.setVisible(True)

PreviewButton = pyxbmct.Button('[COLOR %s][B]Video Preview[/B][/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(PreviewButton,20 , 20, 8, 8)

window.connect(PreviewButton,lambda: buildVideo(Bname))

InstallButton = pyxbmct.Button('[COLOR %s][B]Instalacao Normal[/B][/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(InstallButton,28 , 20, 8, 8)

window.connect(InstallButton, lambda: buildWizard(Bname,'normal'))

FreshStartButton = pyxbmct.Button('[COLOR %s][B]INSTALACAO LIMPA[/B][/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(FreshStartButton,36 , 20, 8, 8)

window.connect(FreshStartButton,lambda: buildWizard(Bname,'fresh'))

ThemeButton = pyxbmct.Button('[COLOR %s][B]Instalar Menu Adulto[/B][/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(ThemeButton,44 , 20, 11, 10)

window.connect(ThemeButton,lambda: buildWizard(Bname,'theme'))

buildthumb = pyxbmct.Image(ICON)

window.placeControl(buildthumb, 21, 30, 45, 19)

buildlist = pyxbmct.List(buttonFocusTexture=LBUTTON)

window.placeControl(buildlist, 14, 1, 79, 15)

buildtextbox = pyxbmct.Label('',textColor='0xFFFFFFFF')

window.placeControl(buildtextbox, 13, 20, 10, 25)

vertextbox = pyxbmct.Label('',textColor='0xFFFFFFFF')

window.placeControl(vertextbox, 60, 20, 10, 15)

koditextbox = pyxbmct.Label('',textColor='0xFFFFFFFF')

window.placeControl(koditextbox, 65, 20, 10, 15)

desctextbox = pyxbmct.TextBox()

window.placeControl(desctextbox, 70, 20, 17, 30)

desctextbox.autoScroll(1100, 1100, 1100)

buildname1 = ADDON.getSetting('buildname')

buildversion1 = ADDON.getSetting('buildversion')

defaultskinname1 = ADDON.getSetting('defaultskinname')

buildtheme1 = ADDON.getSetting('buildtheme')

errors1 = ADDON.getSetting('errors')

lastbuildcheck1 = ADDON.getSetting('lastbuildcheck')

if buildname1 == '' : buildname = None

else:buildname = ADDON.getSetting('buildname')

if buildversion1 == '' : buildversion = None

else:buildversion = ADDON.getSetting('buildversion')

if defaultskinname1 == '' : defaultskinname = None

else:defaultskinname = ADDON.getSetting('defaultskinname')

if buildtheme1 == '' : buildtheme = None

else:buildtheme = ADDON.getSetting('buildtheme')

if errors1 == '' : errors = None

else:errors = ADDON.getSetting('errors')

if lastbuildcheck1 == '' : lastbuildcheck = None

else:lastbuildcheck = ADDON.getSetting('lastbuildcheck')

buildname = pyxbmct.Label('[COLOR %s]Nome da Build Instalada: [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR,buildname))

window.placeControl(buildname, 90, 2, 8, 25)

buildversion = pyxbmct.Label('[COLOR %s]Versao da Build Instalada: [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR,buildversion))

window.placeControl(buildversion, 95, 2, 8, 20)

skinname = pyxbmct.Label('[COLOR %s]Skin da Build: [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR,defaultskinname))

window.placeControl(skinname, 100, 2, 11, 20)

buildtheme = pyxbmct.Label('[COLOR %s]Menu Adulto: [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR,buildtheme))

window.placeControl(buildtheme, 90, 28, 8, 21)

errorinstall = pyxbmct.Label('[COLOR %s]Erros Encontrados na Instalacao: [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR,errors))

window.placeControl(errorinstall, 95, 28, 8, 20)

lastupdatchk = pyxbmct.Label('[COLOR %s]Ultima verificacao de UPDATE: [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR,DESCOLOR,lastbuildcheck))

window.placeControl(lastupdatchk, 100, 28, 8, 20)

buildlist.reset()

buildlist.setVisible(True)

buildthumb.setVisible(True)

InstallButton.setVisible(True)

FreshStartButton.setVisible(True)

buildtextbox.setVisible(True)

vertextbox.setVisible(True)

koditextbox.setVisible(True)

desctextbox.setVisible(True)

buildname.setVisible(True)

buildversion.setVisible(True)

buildtheme.setVisible(True)

skinname.setVisible(True)

errorinstall.setVisible(True)

lastupdatchk.setVisible(True)

buildthumb.setImage(ICON)

link = net.http\_GET(BUILDFILE).content.replace('\n','').replace('\r','')

match = re.compile('name="(.+?)"').findall(link)

for name in match:

name = '[COLOR %s]' % LIST\_TEXT +name+'[/COLOR]'

buildlist.addItem(name)

BuildsButton.controlUp(buildlist)

BuildsButton.controlDown(buildlist)

buildlist.controlRight(PreviewButton)

buildlist.controlUp(BuildsButton)

PreviewButton.controlDown(InstallButton)

PreviewButton.controlUp(BuildsButton)

PreviewButton.controlLeft(buildlist)

InstallButton.controlDown(FreshStartButton)

InstallButton.controlUp(PreviewButton)

InstallButton.controlLeft(buildlist)

FreshStartButton.controlDown(ThemeButton)

FreshStartButton.controlUp(InstallButton)

FreshStartButton.controlLeft(buildlist)

ThemeButton.controlUp(FreshStartButton)

ThemeButton.controlLeft(buildlist)

else:

no\_txt.setVisible(True)

wiz.FTGlog('No Build txt')

def AddonInstall(name):

link = wiz.openURL(ADDONFILE).replace('\n','').replace('\r','').replace('\t','').replace('repository=""', 'repository="none"').replace('repositoryurl=""', 'repositoryurl="http://"').replace('repositoryxml=""', 'repositoryxml="http://"')

match = re.compile('name="%s".+?lugin="(.+?)".+?rl="(.+?)".+?epository="(.+?)".+?epositoryxml="(.+?)".+?epositoryurl="(.+?)".+?con="(.+?)".+?anart="(.+?)".+?dult="(.+?)".+?escription="(.+?)"' % name).findall(link)

if len(match) > 0:

x = 0

for plugin, aurl, repository, repositoryxml, repositoryurl, icon, fanart, adult, description in match:

if plugin.lower() == 'skin':

skinInstaller(name, url)

elif plugin.lower() == 'pack':

packInstaller(name, url)

else:

addonInstaller(plugin, url)

def APKinstall(name):

apk = name

url = checkAPK(name, 'url')

apkInstaller1(apk, url)

def AddonList():

global InstallButtonA

global addonlist

global addthumb

global desctextboxA

global addtextbox

global no\_txt

global url

global name

HIDEALL()

if not ADDONFILE == 'http://' and not ADDONFILE == '':

listbgA.setVisible(True)

buildbgA.setVisible(True)

#buttons/objects

InstallButtonA = pyxbmct.Button('[COLOR %s][B]Instalar ADDON[/B][/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(InstallButtonA,30 , 20, 9, 8)

window.connect(InstallButtonA, lambda:AddonInstall(name))#addonInstaller(plugin, url))

addthumb=pyxbmct.Image(ICON)

window.placeControl(addthumb, 31, 30, 45, 19)

addonlist = pyxbmct.List(buttonFocusTexture=LBUTTON)

window.placeControl(addonlist, 24, 1, 79, 15)

addtextbox = pyxbmct.Label('',textColor='0xFFFFFFFF')

window.placeControl(addtextbox, 24, 20, 10, 25)

desctextboxA = pyxbmct.TextBox()

window.placeControl(desctextboxA, 80, 20, 17, 30)

desctextboxA.autoScroll(1100, 1100, 1100)

addonlist.reset()

addonlist.setVisible(True)

addthumb.setVisible(True)

AddonButton.setVisible(True)

APKButton.setVisible(True)

ROMButton.setVisible(True)

EmuButton.setVisible(True)

InstallButtonA.setVisible(True)

addtextbox.setVisible(True)

desctextboxA.setVisible(True)

addthumb.setImage(ICON)

link = net.http\_GET(ADDONFILE).content.replace('\n','').replace('\r','')

match = re.compile('name="(.+?)"').findall(link)

for name in match:

name = '[COLOR %s]' % LIST\_TEXT +name+'[/COLOR]'

addonlist.addItem(name)

AddonButton.controlDown(addonlist)

addonlist.controlRight(InstallButtonA)

addonlist.controlUp(AddonButton)

InstallButtonA.controlLeft(addonlist)

InstallButtonA.controlUp(AddonButton)

else:

no\_txt.setVisible(True)

AddonButton.setVisible(True)

APKButton.setVisible(True)

ROMButton.setVisible(True)

EmuButton.setVisible(True)

wiz.FTGlog('No Addon txt')

def APKList():

global InstallButtonAPK

global apklist

global apkthumb

global desctextboxAPK

global apktextbox

global no\_txt

HIDEALL()

if not APKFILE == 'http://' and not APKFILE == '':

listbgA.setVisible(True)

buildbgA.setVisible(True)

InstallButtonAPK = pyxbmct.Button('[COLOR %s][B]Install[/B][/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(InstallButtonAPK,30 , 20, 9, 8)

window.connect(InstallButtonAPK, lambda:apkInstaller1(name, url))

apkthumb=pyxbmct.Image(ICON)

window.placeControl(apkthumb, 31, 30, 45, 19)

apklist = pyxbmct.List(buttonFocusTexture=LBUTTON)

window.placeControl(apklist, 24, 1, 79, 15)

apktextbox = pyxbmct.Label('',textColor='0xFFFFFFFF')

window.placeControl(apktextbox, 24, 20, 10, 25)

desctextboxAPK = pyxbmct.TextBox()

window.placeControl(desctextboxAPK, 80, 20, 17, 30)

desctextboxAPK.autoScroll(1100, 1100, 1100)

AddonButton.setVisible(True)

APKButton.setVisible(True)

ROMButton.setVisible(True)

EmuButton.setVisible(True)

apklist.reset()

apklist.setVisible(True)

apkthumb.setVisible(True)

InstallButtonAPK.setVisible(True)

apktextbox.setVisible(True)

desctextboxAPK.setVisible(True)

apkthumb.setImage(ICON)

link = net.http\_GET(APKFILE).content.replace('\n','').replace('\r','')

match = re.compile('name="(.+?)"').findall(link)

for name in match:

name = '[COLOR %s]' % LIST\_TEXT +name+'[/COLOR]'

apklist.addItem(name)

APKButton.controlDown(apklist)

apklist.controlRight(InstallButtonAPK)

apklist.controlUp(APKButton)

InstallButtonAPK.controlLeft(apklist)

InstallButtonAPK.controlUp(APKButton)

else:

no\_txt.setVisible(True)

AddonButton.setVisible(True)

APKButton.setVisible(True)

ROMButton.setVisible(True)

EmuButton.setVisible(True)

wiz.FTGlog('No APK txt')

def EmuList():

global emulist

global emuthumb

global desctextboxEMU

global emutextbox

global InstallButtonEMU

global no\_txt

HIDEALL()

if not EMUAPKS == 'http://' and not EMUAPKS == '':

listbgA.setVisible(True)

buildbgA.setVisible(True)

InstallButtonEMU = pyxbmct.Button('[COLOR %s][B]Install[/B][/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(InstallButtonEMU,30 , 20, 9, 8)

window.connect(InstallButtonEMU, lambda:apkInstaller1(name, url))

emulist = pyxbmct.List(buttonFocusTexture=LBUTTON)

window.placeControl(emulist, 24, 1, 79, 15)

emuthumb=pyxbmct.Image(ICON)

window.placeControl(emuthumb, 31, 30, 45, 19)

emutextbox = pyxbmct.Label('',textColor='0xFFFFFFFF')

window.placeControl(emutextbox, 24, 20, 10, 25)

desctextboxEMU = pyxbmct.TextBox()

window.placeControl(desctextboxEMU, 80, 20, 17, 30)

desctextboxEMU.autoScroll(1100, 1100, 1100)

AddonButton.setVisible(True)

APKButton.setVisible(True)

ROMButton.setVisible(True)

EmuButton.setVisible(True)

emulist.setVisible(True)

desctextboxEMU.setVisible(True)

emutextbox.setVisible(True)

InstallButtonEMU.setVisible(True)

emulist.reset()

emulist.setVisible(True)

emuthumb.setVisible(True)

link = net.http\_GET(EMUAPKS).content.replace('\n','').replace('\r','')

match = re.compile('name="(.+?)"').findall(link)

for name in match:

name = '[COLOR %s]' % LIST\_TEXT +name+'[/COLOR]'

emulist.addItem(name)

EmuButton.controlDown(emulist)

emulist.controlRight(InstallButtonEMU)

emulist.controlUp(EmuButton)

InstallButtonEMU.controlLeft(emulist)

InstallButtonEMU.controlUp(EmuButton)

else:

no\_txt.setVisible(True)

AddonButton.setVisible(True)

APKButton.setVisible(True)

ROMButton.setVisible(True)

EmuButton.setVisible(True)

wiz.FTGlog('No EMU txt')

def RomList():

global romlist

global romthumb

global desctextboxROM

global romtextbox

global InstallButtonROM

global no\_txt

HIDEALL()

if not ROMPACK == 'http://' and not ROMPACK == '':

listbgA.setVisible(True)

buildbgA.setVisible(True)

InstallButtonROM = pyxbmct.Button('[COLOR %s][B]Install[/B][/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(InstallButtonROM,30 , 20, 9, 8)

window.connect(InstallButtonROM, lambda:UNZIPROM())

romlist = pyxbmct.List(buttonFocusTexture=LBUTTON)

window.placeControl(romlist, 24, 1, 79, 15)

romthumb=pyxbmct.Image(ICON)

window.placeControl(romthumb, 31, 30, 45, 19)

romtextbox = pyxbmct.Label('',textColor='0xFFFFFFFF')

window.placeControl(romtextbox, 24, 20, 10, 25)

desctextboxROM = pyxbmct.TextBox()

window.placeControl(desctextboxROM, 80, 20, 17, 30)

desctextboxROM.autoScroll(1100, 1100, 1100)

AddonButton.setVisible(True)

APKButton.setVisible(True)

ROMButton.setVisible(True)

EmuButton.setVisible(True)

romlist.setVisible(True)

romthumb.setVisible(True)

desctextboxROM.setVisible(True)

romtextbox.setVisible(True)

InstallButtonROM.setVisible(True)

romlist.reset()

romlist.setVisible(True)

link = net.http\_GET(ROMPACK).content.replace('\n','').replace('\r','')

match = re.compile('name="(.+?)"').findall(link)

for name in match:

name = '[COLOR %s]' % LIST\_TEXT +name+'[/COLOR]'

romlist.addItem(name)

ROMButton.controlDown(romlist)

romlist.controlRight(InstallButtonROM)

romlist.controlUp(ROMButton)

InstallButtonROM.controlLeft(romlist)

InstallButtonROM.controlUp(ROMButton)

else:

no\_txt.setVisible(True)

AddonButton.setVisible(True)

APKButton.setVisible(True)

ROMButton.setVisible(True)

EmuButton.setVisible(True)

wiz.FTGlog('No ROM txt')

def Un\_Hide\_Net():

mac,inter\_ip,ip,city,state,country,isp = wiz.net\_info()

MAC.setLabel('[COLOR %s]Mac:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, mac))

INTER\_IP.setLabel('[COLOR %s]IP Interno:[COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR,inter\_ip))

IP.setLabel('[COLOR %s]IP Externo:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR,ip))

CITY.setLabel('[COLOR %s]Cidade:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR,city))

STATE.setLabel('[COLOR %s]Estado:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR,state))

COUNTRY.setLabel('[COLOR %s]Pais:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR,country))

ISP.setLabel('[COLOR %s]Provedor:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR,isp))

def Maint():

HIDEALL()

global total\_clean\_button

global total\_cache\_button

global total\_packages\_button

global total\_thumbnails\_button

global TC

global DC

global DPK

global DTH

global TPK

global TTH

global speedtest\_button

global sysinfo\_title

global version1

global store

global rom\_used

global rom\_free

global rom\_total

global mem

global ram\_used

global ram\_free

global ram\_total

global kodi

global total

global video

global program

global music

global picture

global repos

global skins

global scripts

sysinfobg.setVisible(True)

maintbg.setVisible(True)

speedthumb.setVisible(True)

netinfobg.setVisible(True)

sizepack = wiz.getSize(PACKAGES)

totalpack = wiz.getTotal(PACKAGES)

sizethumb = wiz.getSize(THUMBS)

totalthumb = wiz.getTotal(THUMBS)

sizecache = wiz.getCacheSize()

totalsize = sizepack+sizethumb+sizecache

picture, music, video, programs, repos, scripts, skins, codename, version, name,storage\_free ,storage\_used, storage\_total, ram\_free, ram\_used, ram\_total = wiz.SYSINFO()

##sysinfo

sysinfo\_title = pyxbmct.Label('[COLOR %s][B]INFORMACAO DO SISTEMA[/B][/COLOR]' % DES\_T\_COLOR)

window.placeControl(sysinfo\_title, 12, 31, 10, 15)

version1 = pyxbmct.Label('[COLOR %s]Versao Atual do Kodi:[/COLOR] [COLOR %s]%s[/COLOR] - [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, codename, DESCOLOR, version))

window.placeControl(version1, 18, 35, 10, 15)

store = pyxbmct.Label('[B][COLOR %s]INFORMACAO DE ARMAZENAMENTO[/COLOR][/B]'% DESCOLOR)

window.placeControl(store, 23, 34, 10, 15)

rom\_used=pyxbmct.Label('[COLOR %s]Total Usado:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, storage\_free))

window.placeControl(rom\_used, 28, 39, 10, 10)

rom\_free=pyxbmct.Label('[COLOR %s]Total Livre:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, storage\_used))

window.placeControl(rom\_free, 32, 39, 10, 10)

rom\_total=pyxbmct.Label('[COLOR %s]Espaco Total:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, storage\_total))

window.placeControl(rom\_total, 37, 39, 10, 10)

mem = pyxbmct.Label('[B][COLOR %s]INFORMACOES DA MEMORIA[/COLOR][/B]' % DESCOLOR)

window.placeControl(mem, 43, 36, 10, 12)

### Hello, how are you

ram\_used=pyxbmct.Label('[COLOR %s]Total Usado:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, ram\_used))

window.placeControl(ram\_used, 48, 39, 10, 10)

ram\_free=pyxbmct.Label('[COLOR %s]Total Disponivel:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, ram\_free))

window.placeControl(ram\_free, 53, 39, 10, 10)

ram\_total=pyxbmct.Label('[COLOR %s]Memoria Total:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, ram\_total))

window.placeControl(ram\_total, 58, 39, 10, 10)

##addon info

kodi = pyxbmct.Label('[COLOR %s]Name:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, name))

window.placeControl(kodi, 17, 22, 10, 15)

totalcount = len(picture) + len(music) + len(video) + len(programs) + len(scripts) + len(skins) + len(repos)

total = pyxbmct.Label('[COLOR %s]Total de Addons: [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, totalcount))

window.placeControl(total, 22, 22, 10, 10)

video=pyxbmct.Label('[COLOR %s]Addons de Video:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, str(len(video))))

window.placeControl(video, 27, 22, 10, 10)

program=pyxbmct.Label('[COLOR %s]Addons de Programas:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, str(len(programs))))

window.placeControl(program, 33, 22, 10, 10)

music=pyxbmct.Label('[COLOR %s]Addons de Musica:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, str(len(music))))

window.placeControl(music, 37, 22, 10, 10)

picture=pyxbmct.Label('[COLOR %s]Addons de Imagens:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, str(len(picture))))

window.placeControl(picture, 42, 22, 10, 10)

repos=pyxbmct.Label('[COLOR %s]Repositorios:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, str(len(repos))))

window.placeControl(repos, 47, 22, 10, 10)

skins=pyxbmct.Label('[COLOR %s]Skins Instaladas: [/COLOR][COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, str(len(skins))))

window.placeControl(skins, 52, 22, 10, 10)

scripts=pyxbmct.Label('[COLOR %s]Scripts/Modulos:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR, str(len(scripts))))

window.placeControl(scripts, 57, 22, 10, 10)

global MAC

global INTER\_IP

global IP

global CITY

global STATE

global COUNTRY

global ISP

global netinfo\_title

global un\_hide\_net

global settings\_button1

global trigger\_title

###NET INFO

netinfo\_title = pyxbmct.Label('[COLOR %s][B]INFORMACOES DE SUA REDE[/B][/COLOR]'% DES\_T\_COLOR)

window.placeControl(netinfo\_title, 12, 7, 10, 20)

MAC = pyxbmct.Label('[COLOR %s]Mac:[/COLOR] [COLOR %s]Oculto[/COLOR]' % (DES\_T\_COLOR, DESCOLOR))

window.placeControl(MAC, 18 , 1 , 10, 18)

INTER\_IP = pyxbmct.Label('[COLOR %s]IP Interno: [COLOR %s]Oculto[/COLOR]' % (DES\_T\_COLOR, DESCOLOR))

window.placeControl(INTER\_IP, 23 , 1 , 10, 18)

IP = pyxbmct.Label('[COLOR %s]IP Externo:[/COLOR] [COLOR %s]Oculto[/COLOR]' % (DES\_T\_COLOR, DESCOLOR))

window.placeControl(IP, 28 , 1 , 10, 18)

CITY = pyxbmct.Label('[COLOR %s]Cidade:[/COLOR] [COLOR %s]Oculto[/COLOR]' % (DES\_T\_COLOR, DESCOLOR))

window.placeControl(CITY, 33 , 1 , 10, 18)

STATE = pyxbmct.Label('[COLOR %s]Estado:[/COLOR] [COLOR %s]Oculto[/COLOR]' % (DES\_T\_COLOR, DESCOLOR))

window.placeControl(STATE, 38 , 1 , 10, 18)

COUNTRY = pyxbmct.Label('[COLOR %s]Pais:[/COLOR] [COLOR %s]Oculto[/COLOR]' % (DES\_T\_COLOR, DESCOLOR))

window.placeControl(COUNTRY, 43 , 1 , 10, 18)

ISP = pyxbmct.Label('[COLOR %s]Provedor:[/COLOR] [COLOR %s]Oculto[/COLOR]' % (DES\_T\_COLOR, DESCOLOR))

window.placeControl(ISP, 48 , 1 , 10, 18)

#maint lables#

TC = pyxbmct.Label('[COLOR %s]Tamanho:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR,wiz.convertSize(totalsize)))

window.placeControl(TC, 80 , 21 , 10, 9)

DC = pyxbmct.Label('[COLOR %s]Tamanho:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR,wiz.convertSize(sizecache)))

window.placeControl(DC, 96 , 21 , 10, 9)

DPK = pyxbmct.Label('[COLOR %s]Tamanho:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR,wiz.convertSize(sizepack)))

window.placeControl(DPK, 96 , 31 , 10, 9)

TPK = pyxbmct.Label('[COLOR %s]Arquivos:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR,totalpack))

window.placeControl(TPK, 101 , 31 , 10, 9)

DTH = pyxbmct.Label('[COLOR %s]Tamanho:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR,wiz.convertSize(sizethumb)))

window.placeControl(DTH, 96 , 41 , 10, 9)

TTH = pyxbmct.Label('[COLOR %s]Arquivos:[/COLOR] [COLOR %s]%s[/COLOR]' % (DES\_T\_COLOR, DESCOLOR,totalthumb))

window.placeControl(TTH, 101 , 41 , 10, 9)

#buttons#

total\_clean\_button = pyxbmct.Button('[COLOR %s]Limpeza Total[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(total\_clean\_button, 72 , 21 , 9, 8)

window.connect(total\_clean\_button,lambda: totalClean())

### I'm good, u?

total\_cache\_button = pyxbmct.Button('[COLOR %s]Apagar CACHE[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(total\_cache\_button, 87 , 21 , 9, 9)

window.connect(total\_cache\_button,lambda: clearCache())

total\_packages\_button = pyxbmct.Button('[COLOR %s]Apagar TEMPORARIOS[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(total\_packages\_button, 87 , 31 , 9, 10)

window.connect(total\_packages\_button,lambda: clearPackages())

total\_thumbnails\_button = pyxbmct.Button('[COLOR %s]Apagar MINIATURAS[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(total\_thumbnails\_button, 87 , 41 , 9, 9)

window.connect(total\_thumbnails\_button, lambda: clearThumb(type=None))

speedtest\_button = pyxbmct.Button('[COLOR %s]Speed Test[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(speedtest\_button, 58 , 1 , 9, 8)

window.connect(speedtest\_button, lambda: runspeedtest())

un\_hide\_net = pyxbmct.Button('[COLOR %s]Minhas Informacoes Rede[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(un\_hide\_net, 58, 9, 9, 11)

window.connect(un\_hide\_net, lambda: Un\_Hide\_Net())

trigger\_title = pyxbmct.Label('[COLOR %s]Manutencao Geral.[/COLOR]'% DES\_T\_COLOR)

window.placeControl(trigger\_title, 72 , 30 , 1, 11)

settings\_button1 = pyxbmct.Button('[COLOR %s]Configuracoes[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(settings\_button1, 72 , 41 , 9, 8)

window.connect(settings\_button1, lambda: wiz.openS('Maintenance'))

#HIDEALL()

splash.setVisible(False)

total\_clean\_button.setVisible(True)

total\_cache\_button.setVisible(True)

total\_packages\_button.setVisible(True)

total\_thumbnails\_button.setVisible(True)

TC.setVisible(True)

DC.setVisible(True)

DPK.setVisible(True)

DTH.setVisible(True)

TPK.setVisible(True)

TTH.setVisible(True)

speedtest\_button.setVisible(True)

sysinfo\_title.setVisible(True)

version1.setVisible(True)

store.setVisible(True)

rom\_used.setVisible(True)

rom\_free.setVisible(True)

rom\_total.setVisible(True)

mem.setVisible(True)

ram\_used.setVisible(True)

ram\_free.setVisible(True)

ram\_total.setVisible(True)

kodi.setVisible(True)

total.setVisible(True)

video.setVisible(True)

program.setVisible(True)

music.setVisible(True)

picture.setVisible(True)

repos.setVisible(True)

skins.setVisible(True)

scripts.setVisible(True)

netinfo\_title.setVisible(True)

MAC.setVisible(True)

un\_hide\_net.setVisible(True)

INTER\_IP.setVisible(True)

IP.setVisible(True)

ISP.setVisible(True)

CITY.setVisible(True)

STATE.setVisible(True)

COUNTRY.setVisible(True)

settings\_button1.setVisible(True)

trigger\_title.setVisible(True)

MaintButton.controlDown(speedtest\_button)

speedtest\_button.controlUp(MaintButton)

speedtest\_button.controlDown(total\_clean\_button)

speedtest\_button.controlRight(un\_hide\_net)

un\_hide\_net.controlUp(MaintButton)

un\_hide\_net.controlDown(total\_clean\_button)

un\_hide\_net.controlRight(total\_clean\_button)

un\_hide\_net.controlLeft(speedtest\_button)

total\_clean\_button.controlUp(MaintButton)

total\_clean\_button.controlDown(total\_cache\_button)

total\_clean\_button.controlRight(settings\_button1)

total\_clean\_button.controlLeft(un\_hide\_net)

total\_cache\_button.controlUp(total\_clean\_button)

total\_cache\_button.controlRight(total\_packages\_button)

total\_cache\_button.controlLeft(un\_hide\_net)

total\_packages\_button.controlUp(settings\_button1)

total\_packages\_button.controlRight(total\_thumbnails\_button)

total\_packages\_button.controlLeft(total\_cache\_button)

total\_thumbnails\_button.controlUp(settings\_button1)

total\_thumbnails\_button.controlLeft(total\_packages\_button)

settings\_button1.controlUp(MaintButton)

settings\_button1.controlDown(total\_thumbnails\_button)

settings\_button1.controlLeft(total\_clean\_button)

def BackRes():

HIDEALL()

global favs

global backuploc

global Backup

global backup\_build\_button

global backup\_gui\_button

global backup\_addondata\_button

global restore\_build\_button

global restore\_gui\_button

global restore\_addondata\_button

global clear\_backup\_button

global savefav\_button

global restorefav\_button

global clearfav\_button

global backupaddonpack\_button

global restore\_title

global delete\_title

global set\_title

global settings\_button

bakresbg.setVisible(True)

last = str(FAVSsave) if not FAVSsave == '' else 'Seus Favoritos nao foi salvo ainda.'

#loc info#

favs = pyxbmct.Label('[B][COLOR %s]Ultima Gravacao:[/COLOR] [COLOR %s]%s[/COLOR][/B]' % (DES\_T\_COLOR, DESCOLOR,str(last)))

window.placeControl(favs, 14, 3, 10, 30)

backuploc = pyxbmct.Label('[B][COLOR %s]Local do Back-up: [COLOR %s]%s[/COLOR][/B]' % (DES\_T\_COLOR, DESCOLOR,MYBUILDS))

window.placeControl(backuploc, 22, 3, 10, 30)

#backup#

Backup = pyxbmct.Label('[B][COLOR %s]FERRAMENTAS DE BACKUP:[/COLOR][/B]' % DES\_T\_COLOR)

window.placeControl(Backup, 32, 3, 10, 11)

backup\_build\_button = pyxbmct.Button('[COLOR %s] Build Completa[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(backup\_build\_button, 42, 3, 10, 10)

window.connect(backup\_build\_button,lambda: wiz.backUpOptions('build'))

backup\_gui\_button = pyxbmct.Button('[COLOR %s] GUI[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(backup\_gui\_button, 52, 3, 10, 10)

window.connect(backup\_gui\_button,lambda: wiz.backUpOptions('guifix'))

backupaddonpack\_button = pyxbmct.Button('[COLOR %s] Salvar Addons[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(backupaddonpack\_button, 62, 3, 10, 10)

window.connect(backupaddonpack\_button,lambda: wiz.backUpOptions('addon pack'))

# I'm ok, thanks for asking

backup\_addondata\_button = pyxbmct.Button('[COLOR %s] Salvar Addons Data[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(backup\_addondata\_button, 72, 3, 10, 10)

window.connect(backup\_addondata\_button,lambda: wiz.backUpOptions('addondata'))

savefav\_button = pyxbmct.Button('[COLOR %s] Salvar Favoritos[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(savefav\_button, 82, 3, 10, 10)

window.connect(savefav\_button,lambda: wiz.BACKUPFAV())

#restore#

restore\_title = pyxbmct.Label('[B][COLOR %s]FERRAMENTAS DE RESTAURACAO:[/COLOR][/B]' % DES\_T\_COLOR)

window.placeControl(restore\_title, 32, 20, 10, 12)

restore\_build\_button = pyxbmct.Button('[COLOR %s]Restaurar Build[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(restore\_build\_button, 42, 20, 10, 12)

window.connect(restore\_build\_button,lambda: restoreit('build'))

restore\_gui\_button = pyxbmct.Button('[COLOR %s]GUI[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(restore\_gui\_button, 52, 20, 10, 10)

window.connect(restore\_gui\_button,lambda: restoreit('gui'))

### Nope not here

restore\_addondata\_button = pyxbmct.Button('[COLOR %s]Restaurar Addon Data[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(restore\_addondata\_button, 62, 20, 10, 10)

window.connect(restore\_addondata\_button,lambda: restoreit('addondata'))

restorefav\_button = pyxbmct.Button('[COLOR %s]Restaurar Favoritos[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(restorefav\_button, 72, 20, 10, 10)

window.connect(restorefav\_button,lambda: wiz.RESTOREFAV())

#clear backups#

delete\_title = pyxbmct.Label('[B][COLOR red]Ferramentas de limpeza:[/COLOR][/B]')

window.placeControl(delete\_title, 54, 37, 10, 10)

clearfav\_button = pyxbmct.Button('[COLOR %s]Apagar Favoritos[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(clearfav\_button, 64, 37, 10, 10)

window.connect(clearfav\_button,lambda: wiz.DELFAV())

clear\_backup\_button = pyxbmct.Button('[COLOR %s]Limpar pasta de Backup[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(clear\_backup\_button, 74, 37, 10, 11)

window.connect(clear\_backup\_button,lambda: wiz.cleanupBackup())

#settings

set\_title = pyxbmct.Label('[B][COLOR %s]CONFIGURACOES:[/COLOR][/B]' % DES\_T\_COLOR)

window.placeControl(set\_title, 32, 37, 10, 10)

settings\_button = pyxbmct.Button('[COLOR %s]Configuracoes[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(settings\_button, 42, 37, 10, 10)

window.connect(settings\_button, lambda: wiz.openS('Maintenance'))

favs.setVisible(True)

backuploc.setVisible(True)

Backup.setVisible(True)

backup\_build\_button.setVisible(True)

backup\_gui\_button.setVisible(True)

backup\_addondata\_button.setVisible(True)

restore\_build\_button.setVisible(True)

restore\_gui\_button.setVisible(True)

restore\_addondata\_button.setVisible(True)

clear\_backup\_button.setVisible(True)

savefav\_button.setVisible(True)

restorefav\_button.setVisible(True)

clearfav\_button.setVisible(True)

backupaddonpack\_button.setVisible(True)

restore\_title.setVisible(True)

delete\_title.setVisible(True)

set\_title.setVisible(True)

settings\_button.setVisible(True)

BackResButton.controlDown(backup\_build\_button)

backup\_build\_button.controlUp(BackResButton)

backup\_build\_button.controlDown(backup\_gui\_button)

backup\_build\_button.controlRight(restore\_build\_button)

backup\_gui\_button.controlUp(backup\_build\_button)

backup\_gui\_button.controlDown(backupaddonpack\_button)

backup\_gui\_button.controlRight(restore\_gui\_button)

backupaddonpack\_button.controlUp(backup\_gui\_button)

backupaddonpack\_button.controlDown(backup\_addondata\_button)

backupaddonpack\_button.controlRight(restore\_addondata\_button)

backup\_addondata\_button.controlUp(backupaddonpack\_button)

backup\_addondata\_button.controlDown(savefav\_button)

backup\_addondata\_button.controlRight(restorefav\_button)

savefav\_button.controlUp(backup\_addondata\_button)

savefav\_button.controlRight(restorefav\_button)

restore\_build\_button.controlUp(BackResButton)

restore\_build\_button.controlDown(restore\_gui\_button)

restore\_build\_button.controlRight(settings\_button)

restore\_build\_button.controlLeft(backup\_build\_button)

restore\_gui\_button.controlUp(restore\_build\_button)

restore\_gui\_button.controlDown(restore\_addondata\_button)

restore\_gui\_button.controlRight(settings\_button)

restore\_gui\_button.controlLeft(backup\_gui\_button)

restore\_addondata\_button.controlUp(restore\_gui\_button)

restore\_addondata\_button.controlDown(restorefav\_button)

restore\_addondata\_button.controlRight(clearfav\_button)

restore\_addondata\_button.controlLeft(backupaddonpack\_button)

restorefav\_button.controlUp(restore\_addondata\_button)

restorefav\_button.controlRight(clear\_backup\_button)

restorefav\_button.controlLeft(backup\_addondata\_button)

settings\_button.controlUp(BackResButton)

settings\_button.controlDown(clearfav\_button)

settings\_button.controlLeft(restore\_build\_button)

clearfav\_button.controlUp(settings\_button)

clearfav\_button.controlDown(clear\_backup\_button)

clearfav\_button.controlLeft(restore\_addondata\_button)

clear\_backup\_button.controlUp(clearfav\_button)

clear\_backup\_button.controlLeft(restorefav\_button)

def Tools():

HIDEALL()

global view\_error\_button

global full\_log\_button

global upload\_log\_button

global Log\_title

global Log\_errors

toolsbg.setVisible(True)

Log\_title = pyxbmct.Label('[B][COLOR %s]Logging Tools[/COLOR][/B]' % DES\_T\_COLOR)

window.placeControl(Log\_title, 15, 4, 10, 10)

Log\_errors = pyxbmct.Label('[COLOR %s][B]Errors in Log:[/B][/COLOR] %s' % (OTHER\_BUTTONS\_TEXT, log\_tools()))

window.placeControl(Log\_errors, 22, 4, 10, 15)

## uhhh not here

view\_error\_button = pyxbmct.Button('[COLOR %s]View Errors[/COLOR]'% OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(view\_error\_button, 31, 4, 9, 9)

window.connect(view\_error\_button, lambda: errorChecking(log=None, count=None, last=None))

full\_log\_button = pyxbmct.Button('[COLOR %s]View Full Log[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(full\_log\_button, 40, 4, 9, 9)

window.connect(full\_log\_button,lambda :LogViewer())

upload\_log\_button = pyxbmct.Button('[COLOR %s]Upload Full Log[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(upload\_log\_button, 49, 4, 9, 9)

window.connect(upload\_log\_button,lambda :uploadLog.Main())

#Addon#

global removeaddons\_button

global removeaddondata\_u\_button

global removeaddondata\_e\_button

global checksources\_button

global checkrepos\_button

global forceupdate\_button

global fixaddonupdate\_button

global removeaddondata\_all\_button

global scan

global fix

global delet

global delet1

global Addon

Addon = pyxbmct.Label('[B][COLOR %s]Addon Tools[/COLOR][/B]' % DES\_T\_COLOR)

window.placeControl(Addon, 69, 21, 9, 9)

#buttons#

scan = pyxbmct.Label('[B][COLOR %s]Scan For:[/COLOR][/B]' % DES\_T\_COLOR)

window.placeControl(scan, 75, 4, 9, 10)

checksources\_button = pyxbmct.Button('[COLOR %s]Broken Sources[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(checksources\_button, 82, 3, 9, 10)

window.connect(checksources\_button,lambda: wiz.checkSources())

checkrepos\_button = pyxbmct.Button('[COLOR %s]Broken Repositories[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(checkrepos\_button, 92, 3, 9, 10)

window.connect(checkrepos\_button,lambda: wiz.checkRepos())

fix = pyxbmct.Label('[B][COLOR %s]Force / Fix:[/COLOR][/B]' % DES\_T\_COLOR)

window.placeControl(fix, 75, 16, 9, 10)

forceupdate\_button = pyxbmct.Button('[COLOR %s]Update Addons[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(forceupdate\_button, 82, 15, 9, 9)

window.connect(forceupdate\_button,lambda: wiz.forceUpdate())

fixaddonupdate\_button = pyxbmct.Button('[COLOR %s]Addons Not Updating[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(fixaddonupdate\_button, 92, 15, 9, 9)

window.connect(fixaddonupdate\_button,lambda: fixaddonupdate)

### I see you looking

delet = pyxbmct.Label('[B][COLOR %s]Delete: [/COLOR][/B]' % DES\_T\_COLOR)

window.placeControl(delet, 75, 28, 9, 10)

removeaddons\_button = pyxbmct.Button('[COLOR %s]Delete Selected Addons[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(removeaddons\_button, 82, 26, 9, 9)

window.connect(removeaddons\_button,lambda: removeAddonMenu)

removeaddondata\_all\_button = pyxbmct.Button('[COLOR %s]All Addon Data[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(removeaddondata\_all\_button, 92, 26, 9, 9)

window.connect(removeaddondata\_all\_button,lambda: removeAddonData('all'))

delet1 = pyxbmct.Label('[B][COLOR %s]Delete: [/COLOR][/B]' % DES\_T\_COLOR)

window.placeControl(delet1, 75, 39, 9, 10)

removeaddondata\_u\_button = pyxbmct.Button('[COLOR %s]Uninstalled Folders[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(removeaddondata\_u\_button, 82, 37, 9, 9)

window.connect(removeaddondata\_u\_button,lambda: removeAddonData('uninstalled'))

removeaddondata\_e\_button = pyxbmct.Button('[COLOR %s]Empty Folders[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(removeaddondata\_e\_button, 92, 37, 9, 9)

window.connect(removeaddondata\_e\_button,lambda: removeAddonData('empty'))

####White List###

global WhiteList

global whitelist\_edit\_button

global whitelist\_view\_button

global whitelist\_clear\_button

global whitelist\_import\_button

global whitelist\_export\_button

WhiteList = pyxbmct.Label('[B][COLOR %s]White List Tools: [/COLOR][/B]' % DES\_T\_COLOR)

window.placeControl(WhiteList, 15, 36, 9, 9)

whitelist\_edit\_button = pyxbmct.Button('[COLOR %s]WhiteList: Edit[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(whitelist\_edit\_button, 22, 36, 9, 9)

window.connect(whitelist\_edit\_button,lambda: wiz.whiteList('edit'))

whitelist\_view\_button = pyxbmct.Button('[COLOR %s]WhiteList: View[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(whitelist\_view\_button, 31, 36, 9, 9)

window.connect(whitelist\_view\_button,lambda: wiz.whiteList('view'))

whitelist\_clear\_button = pyxbmct.Button('[COLOR %s]WhiteList: Clear[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(whitelist\_clear\_button, 40, 36, 9, 9)

window.connect(whitelist\_clear\_button,lambda: wiz.whiteList('clear'))

### It's not here

whitelist\_import\_button = pyxbmct.Button('[COLOR %s]WhiteList: Import[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(whitelist\_import\_button, 49, 36, 9, 9)

window.connect(whitelist\_import\_button,lambda: wiz.whiteList('Import'))

whitelist\_export\_button = pyxbmct.Button('[COLOR %s]WhiteList: Export[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(whitelist\_export\_button, 58, 36, 9, 9)

window.connect(whitelist\_export\_button,lambda: wiz.whiteList('export'))

######## Advanced ########

global Advan

global autoadvanced\_buttonQ

global autoadvanced\_button

global currentsettings\_button

global removeadvanced\_button

Advan = pyxbmct.Label('[B][COLOR %s]Advanced Settings Tools[/COLOR][/B]'% DES\_T\_COLOR)

window.placeControl(Advan, 15, 20, 9, 13)

#buttons#

autoadvanced\_buttonQ = pyxbmct.Button('[COLOR %s]Quick Config[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(autoadvanced\_buttonQ, 22, 21, 9, 9)

window.connect(autoadvanced\_buttonQ,lambda: notify.autoConfig2())

autoadvanced\_button = pyxbmct.Button('[COLOR %s]Full Config[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(autoadvanced\_button, 31, 21, 9, 9)

window.connect(autoadvanced\_button,lambda: notify.autoConfig())

##Wait it's here!!

currentsettings\_button = pyxbmct.Button('[COLOR %s]Current Settings[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(currentsettings\_button, 40, 21, 9, 9)

window.connect(currentsettings\_button,lambda: viewAdvanced())

removeadvanced\_button = pyxbmct.Button('[COLOR %s]Delete Settings[/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(removeadvanced\_button, 49, 21, 9, 9)

window.connect(removeadvanced\_button,lambda: removeAdvanced())

view\_error\_button.setVisible(True)

full\_log\_button.setVisible(True)

upload\_log\_button.setVisible(True)

removeaddons\_button.setVisible(True)

removeaddondata\_all\_button.setVisible(True)

removeaddondata\_u\_button.setVisible(True)

removeaddondata\_e\_button.setVisible(True)

checksources\_button.setVisible(True)

checkrepos\_button.setVisible(True)

forceupdate\_button.setVisible(True)

fixaddonupdate\_button.setVisible(True)

Log\_title.setVisible(True)

Log\_errors.setVisible(True)

Addon.setVisible(True)

scan.setVisible(True)

fix.setVisible(True)

delet.setVisible(True)

delet1.setVisible(True)

WhiteList.setVisible(True)

whitelist\_edit\_button.setVisible(True)

whitelist\_view\_button.setVisible(True)

whitelist\_clear\_button.setVisible(True)

whitelist\_import\_button.setVisible(True)

whitelist\_export\_button.setVisible(True)

Advan.setVisible(True)

autoadvanced\_buttonQ.setVisible(True)

autoadvanced\_button.setVisible(True)

currentsettings\_button.setVisible(True)

removeadvanced\_button.setVisible(True)

ToolsButton.controlDown(view\_error\_button)

view\_error\_button.controlUp(ToolsButton)

view\_error\_button.controlDown(full\_log\_button)

view\_error\_button.controlRight(autoadvanced\_button)

full\_log\_button.controlUp(view\_error\_button)

full\_log\_button.controlDown(upload\_log\_button)

full\_log\_button.controlRight(currentsettings\_button)

upload\_log\_button.controlUp(full\_log\_button)

upload\_log\_button.controlDown(checksources\_button)

upload\_log\_button.controlRight(removeadvanced\_button)

autoadvanced\_buttonQ.controlUp(ToolsButton)

autoadvanced\_buttonQ.controlLeft(view\_error\_button)

autoadvanced\_buttonQ.controlDown(autoadvanced\_button)

autoadvanced\_buttonQ.controlRight(whitelist\_edit\_button)

autoadvanced\_button.controlUp(autoadvanced\_buttonQ)

autoadvanced\_button.controlLeft(view\_error\_button)

autoadvanced\_button.controlDown(currentsettings\_button)

autoadvanced\_button.controlRight(whitelist\_view\_button)

currentsettings\_button.controlUp(autoadvanced\_button)

currentsettings\_button.controlLeft(full\_log\_button)

currentsettings\_button.controlDown(removeadvanced\_button)

currentsettings\_button.controlRight(whitelist\_clear\_button)

removeadvanced\_button.controlUp(currentsettings\_button)

removeadvanced\_button.controlLeft(upload\_log\_button)

removeadvanced\_button.controlDown(forceupdate\_button)

removeadvanced\_button.controlRight(whitelist\_import\_button)

whitelist\_edit\_button.controlUp(ToolsButton)

whitelist\_edit\_button.controlLeft(autoadvanced\_buttonQ)

whitelist\_edit\_button.controlDown(whitelist\_view\_button)

whitelist\_view\_button.controlUp(whitelist\_edit\_button)

whitelist\_view\_button.controlLeft(autoadvanced\_button)

whitelist\_view\_button.controlDown(whitelist\_clear\_button)

whitelist\_clear\_button.controlUp(whitelist\_view\_button)

whitelist\_clear\_button.controlLeft(currentsettings\_button)

whitelist\_clear\_button.controlDown(whitelist\_import\_button)

whitelist\_import\_button.controlUp(whitelist\_clear\_button)

whitelist\_import\_button.controlLeft(removeadvanced\_button)

whitelist\_import\_button.controlDown(whitelist\_export\_button)

whitelist\_export\_button.controlUp(whitelist\_import\_button)

whitelist\_export\_button.controlLeft(removeadvanced\_button)

whitelist\_export\_button.controlDown(removeaddondata\_u\_button)

checksources\_button.controlUp(upload\_log\_button)

checksources\_button.controlDown(checkrepos\_button)

checksources\_button.controlRight(forceupdate\_button)

checkrepos\_button.controlUp(checksources\_button)

checkrepos\_button.controlRight(fixaddonupdate\_button)

forceupdate\_button.controlUp(removeadvanced\_button)

forceupdate\_button.controlLeft(checksources\_button)

forceupdate\_button.controlDown(fixaddonupdate\_button)

forceupdate\_button.controlRight(removeaddons\_button)

fixaddonupdate\_button.controlUp(forceupdate\_button)

fixaddonupdate\_button.controlLeft(checkrepos\_button)

fixaddonupdate\_button.controlRight(removeaddondata\_all\_button)

removeaddons\_button.controlUp(removeadvanced\_button)

removeaddons\_button.controlLeft(forceupdate\_button)

removeaddons\_button.controlDown(removeaddondata\_all\_button)

removeaddons\_button.controlRight(removeaddondata\_u\_button)

removeaddondata\_all\_button.controlUp(removeaddons\_button)

removeaddondata\_all\_button.controlLeft(fixaddonupdate\_button)

removeaddondata\_all\_button.controlRight(removeaddondata\_e\_button)

removeaddondata\_u\_button.controlUp(whitelist\_export\_button)

removeaddondata\_u\_button.controlLeft(removeaddons\_button)

removeaddondata\_u\_button.controlDown(removeaddondata\_e\_button)

removeaddondata\_e\_button.controlUp(removeaddondata\_u\_button)

removeaddondata\_e\_button.controlLeft(removeaddondata\_all\_button)

def Installables():

global AddonButton

global APKButton

global ROMButton

global EmuButton

HIDEALL()

listbgA.setVisible(True)

buildbgA.setVisible(True)

AddonButton = pyxbmct.Button('[COLOR %s][B]ADDONS[/B][/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=EXIT,noFocusTexture=BUTTON)

window.placeControl(AddonButton,12 , 9, 7, 8)

window.connect(AddonButton, lambda: AddonList())

APKButton = pyxbmct.Button('[COLOR %s][B]-[/B][/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=EXIT,noFocusTexture=BUTTON)

window.placeControl(APKButton,12 , 17 , 7, 8)

window.connect(APKButton, lambda: APKList())

ROMButton = pyxbmct.Button('[COLOR %s][B]-[/B][/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=EXIT,noFocusTexture=BUTTON)

window.placeControl(ROMButton,12 , 25 , 7, 8)

window.connect(ROMButton, lambda: RomList())

EmuButton = pyxbmct.Button('[COLOR %s][B]-[/B][/COLOR]' % OTHER\_BUTTONS\_TEXT,focusTexture=EXIT,noFocusTexture=BUTTON)

window.placeControl(EmuButton,12 , 33 , 7, 8)

window.connect(EmuButton, lambda: EmuList())

InstallablesButton.controlDown(AddonButton)

AddonButton.controlUp(InstallablesButton)

APKButton.controlUp(InstallablesButton)

ROMButton.controlUp(InstallablesButton)

EmuButton.controlUp(InstallablesButton)

AddonButton.controlRight(APKButton)

AddonButton.controlLeft(EmuButton)

APKButton.controlRight(ROMButton)

APKButton.controlLeft(AddonButton)

ROMButton.controlRight(EmuButton)

ROMButton.controlLeft(APKButton)

EmuButton.controlRight(AddonButton)

EmuButton.controlLeft(ROMButton)

window.connectEventList(

[pyxbmct.ACTION\_MOVE\_DOWN,

pyxbmct.ACTION\_MOVE\_UP,

pyxbmct.ACTION\_MOUSE\_WHEEL\_DOWN,

pyxbmct.ACTION\_MOUSE\_WHEEL\_UP,

pyxbmct.ACTION\_MOUSE\_MOVE],

list\_update)

##globals

global ver

global fan

global listbg

global buildbg

global maintbg

global speedthumb

global sysinfobg

global netinfobg

global splash

global bakresbg

global toolsbg

global wizinfogb

global listbgA

global buildbgA

global buildinfobg

##################window##########################

window.setGeometry(1280, 720, 100, 50)# Create Window (width,height,rows,cols)

fan=pyxbmct.Image(MAINBG)

window.placeControl(fan, -10, -6, 125, 62)

wiz.FTGlog('Window Opened')

##########Foreground Leave here####################

listbg = pyxbmct.Image(LISTBG)

window.placeControl(listbg, 10, 0, 80, 17)

buildbg = pyxbmct.Image(LISTBG)

window.placeControl(buildbg, 10, 16, 80, 35)

buildinfobg = pyxbmct.Image(LISTBG)

window.placeControl(buildinfobg, 88, 0, 23, 50)

listbgA = pyxbmct.Image(LISTBG)

window.placeControl(listbgA, 20, 0, 80, 17)

buildbgA = pyxbmct.Image(LISTBG)

window.placeControl(buildbgA, 20, 16, 80, 35)

maintbg = pyxbmct.Image(LISTBG)

window.placeControl(maintbg, 70, 19, 40, 32)

sysinfobg = pyxbmct.Image(LISTBG)

window.placeControl(sysinfobg, 10, 19, 60, 32)

netinfobg = pyxbmct.Image(LISTBG)

window.placeControl(netinfobg, 10, 0, 60, 20)

speedthumb = pyxbmct.Image(SpeedBG)

window.placeControl(speedthumb, 70, 0, 40, 20)

splash = pyxbmct.Image(SPLASH)

window.placeControl(splash , 10, 1, 100, 48)

bakresbg = pyxbmct.Image(LISTBG)

window.placeControl(bakresbg , 10, 1, 100, 48)

toolsbg = pyxbmct.Image(LISTBG)

window.placeControl(toolsbg , 10, 1, 100, 48)

wizinfogb = pyxbmct.Image(LISTBG)

window.placeControl(wizinfogb, -6, 9, 9, 32)

wiz\_title = pyxbmct.Label('[COLOR %s][B]%s[/B][/COLOR]' % (uservar.WIZTITLE\_COLOR ,uservar.WIZTITLE))

window.placeControl(wiz\_title, -5, 11, 7, 20)

wiz\_ver = pyxbmct.Label('[COLOR %s]Versao do Instalador: [COLOR %s][B]%s[/B][/COLOR]' % (uservar.VERTITLE\_COLOR,uservar.VER\_NUMBER\_COLOR,VERSION))

window.placeControl(wiz\_ver, -5, 31, 7, 10)

no\_txt = pyxbmct.Image(NOTXT)

window.placeControl(no\_txt, 23, 8, 80, 35)

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

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

##############Main Buttons#############

##globals

global MaintButton

global BackResButton

global ToolsButton

global BuildsButton

global Toolbox

#buttons/objects

BuildsButton= pyxbmct.Button('[COLOR %s][B]Builds[/B][/COLOR]' % MAIN\_BUTTONS\_TEXT, focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(BuildsButton,-2 , 1 , 13, 8)

window.connect(BuildsButton, lambda: BuildList())

MaintButton = pyxbmct.Button('[COLOR %s][B]Manutencao[/B][/COLOR]' % MAIN\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(MaintButton,2 , 9 , 9, 8)

window.connect(MaintButton, lambda: Maint())

BackResButton = pyxbmct.Button('[COLOR %s][B]Backup/Restore[/B][/COLOR]' % MAIN\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(BackResButton,2 , 17, 9, 8)

window.connect(BackResButton, lambda: BackRes())

ToolsButton = pyxbmct.Button('[COLOR %s][B]Ferramentas[/B][/COLOR]' % MAIN\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(ToolsButton,2 , 25, 9, 8)

window.connect(ToolsButton, lambda: Tools())

InstallablesButton = pyxbmct.Button('[COLOR %s][B]Downloads[/B][/COLOR]' % MAIN\_BUTTONS\_TEXT,focusTexture=FBUTTON,noFocusTexture=BUTTON)

window.placeControl(InstallablesButton,2 , 33, 9, 8)

window.connect(InstallablesButton, lambda: Installables())

CloseButton = pyxbmct.Button('[COLOR %s][B]Fechar[/B][/COLOR]' % MAIN\_BUTTONS\_TEXT,focusTexture=EXIT,noFocusTexture=BUTTON)

window.placeControl(CloseButton,-2 , 41,13, 8)

window.connect(CloseButton, window.close)

BuildsButton.controlRight(MaintButton)

BuildsButton.controlLeft(CloseButton)

MaintButton.controlRight(BackResButton)

MaintButton.controlLeft(BuildsButton)

BackResButton.controlRight(ToolsButton)

BackResButton.controlLeft(MaintButton)

ToolsButton.controlRight(InstallablesButton)

ToolsButton.controlLeft(BackResButton)

InstallablesButton.controlRight(CloseButton)

InstallablesButton.controlLeft(ToolsButton)

CloseButton.controlRight(BuildsButton)

CloseButton.controlLeft(InstallablesButton)

############# SOP ###############

# button.controlUp(button)

# button.controlLeft(button)

# button.controlRight(button)

# button.controlDown(button)

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

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

################!!!!!NEVER DELETE!!!!############

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

window.connect(pyxbmct.ACTION\_NAV\_BACK, window.close)

## Event Lists

window.setFocus(BuildsButton)

###Hide everthing initial menu

HIDEALL()

splash.setVisible(True)

###NEEDS WORK

#if wiz.getS('ogwiz') =='true':

# index()

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

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

if mode=='index' : index()

elif mode=='wizardupdate' : wiz.wizardUpdate()

elif mode=='builds' : buildMenu()

elif mode=='viewbuild' : viewBuild(name)

elif mode=='buildinfo' : buildInfo(name)

elif mode=='buildpreview' : buildVideo(name)

elif mode=='install' : buildWizard(name, url)

elif mode=='theme' : buildWizard(name, mode, url)

elif mode=='viewthirdparty' : viewThirdList(name)

elif mode=='installthird' : thirdPartyInstall(name, url)

elif mode=='editthird' : editThirdParty(name); wiz.refresh()

elif mode=='maint' : maintMenu(name)

elif mode=='kodi17fix' : wiz.kodi17Fix()

elif mode=='unknownsources' : skinSwitch.swapUS()

elif mode=='advancedsetting': advancedWindow(name)

elif mode=='autoadvanced1' : showAutoAdvanced1(); wiz.refresh()

elif mode=='removeadvanced' : removeAdvanced(); wiz.refresh()

elif mode=='asciicheck' : wiz.asciiCheck()

elif mode=='backupbuild' : wiz.backUpOptions('build')

elif mode=='backupgui' : wiz.backUpOptions('guifix')

elif mode=='backuptheme' : wiz.backUpOptions('theme')

elif mode=='backupaddonpack': wiz.backUpOptions('addon pack')

elif mode=='backupaddon' : wiz.backUpOptions('addondata')

elif mode=='oldThumbs' : wiz.oldThumbs()

elif mode=='clearbackup' : wiz.cleanupBackup()

elif mode=='convertpath' : wiz.convertSpecial(HOME)

elif mode=='currentsettings': viewAdvanced()

elif mode=='fullclean' : totalClean(); wiz.refresh()

elif mode=='clearcache' : clearCache(); wiz.refresh()

elif mode=='clearpackages' : wiz.clearPackages(); wiz.refresh()

elif mode=='clearcrash' : wiz.clearCrash(); wiz.refresh()

elif mode=='clearthumb' : clearThumb(); wiz.refresh()

elif mode=='cleararchive' : clearArchive(); wiz.refresh()

elif mode=='checksources' : wiz.checkSources(); wiz.refresh()

elif mode=='checkrepos' : wiz.checkRepos(); wiz.refresh()

elif mode=='freshstart' : freshStart()

elif mode=='forceupdate' : wiz.forceUpdate()

elif mode=='forceprofile' : wiz.reloadProfile(wiz.getInfo('System.ProfileName'))

elif mode=='forceclose' : wiz.killxbmc()

elif mode=='forceskin' : wiz.ebi("ReloadSkin()"); wiz.refresh()

elif mode=='hidepassword' : wiz.hidePassword()

elif mode=='unhidepassword' : wiz.unhidePassword()

elif mode=='enableaddons' : enableAddons()

elif mode=='toggleaddon' : wiz.toggleAddon(name, url); wiz.refresh()

elif mode=='togglecache' : toggleCache(name); wiz.refresh()

elif mode=='toggleadult' : wiz.toggleAdult(); wiz.refresh()

elif mode=='changefeq' : changeFeq(); wiz.refresh()

elif mode=='uploadlog' : uploadLog.Main()

elif mode=='viewlog' : LogViewer()

elif mode=='viewwizlog' : LogViewer(WIZLOG)

elif mode=='viewerrorlog' : errorChecking()

elif mode=='viewerrorlast' : errorChecking(last=True)

elif mode=='clearwizlog' : f = open(WIZLOG, 'w'); f.close(); wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), "[COLOR %s]Wizard Log Cleared![/COLOR]" % COLOR2)

elif mode=='purgedb' : purgeDb()

elif mode=='fixaddonupdate' : fixUpdate()

elif mode=='removeaddons' : removeAddonMenu()

elif mode=='removeaddon' : removeAddon(name)

elif mode=='removeaddondata': removeAddonDataMenu()

elif mode=='removedata' : removeAddonData(name)

elif mode=='resetaddon' : total = wiz.cleanHouse(ADDONDATA, ignore=True); wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), "[COLOR %s]Addon\_Data reset[/COLOR]" % COLOR2)

elif mode=='systeminfo' : systemInfo()

elif mode=='restorezip' : restoreit('build')

elif mode=='restoregui' : restoreit('gui')

elif mode=='restoreaddon' : restoreit('addondata')

elif mode=='restoreextzip' : restoreextit('build')

elif mode=='restoreextgui' : restoreextit('gui')

elif mode=='restoreextaddon': restoreextit('addondata')

elif mode=='writeadvanced' : writeAdvanced(name, url)

elif mode=='speedtest' : speedTest()

elif mode=='speed' : speed(); wiz.refresh()# changed from runspeedtest = build conflict

elif mode=='clearspeedtest' : clearSpeedTest(); wiz.refresh()

elif mode=='viewspeedtest' : viewSpeedTest(name); wiz.refresh()

elif mode=='apk' : apkMenu(name, url)

elif mode=='apkscrape' : apkScraper(name)

elif mode=='apkinstall' : apkInstaller(name, url)

elif mode=='apkinstall1' : apkInstaller1(name, url)

elif mode=='rominstall' : romInstaller(name, url)

elif mode=='youtube' : youtubeMenu(name, url)

elif mode=='viewVideo' : playVideo(url)

elif mode=='addons' : addonMenu(name, url)

elif mode=='addonpack' : packInstaller(name, url)

elif mode=='skinpack' : skinInstaller(name, url)

elif mode=='addoninstall' : addonInstaller(name, url)

elif mode=='savedata' : saveMenu()

elif mode=='togglesetting' : wiz.setS(name, 'false' if wiz.getS(name) == 'true' else 'true'); wiz.refresh()

elif mode=='managedata' : manageSaveData(name)

elif mode=='whitelist' : wiz.whiteList(name)

elif mode=='trakt' : traktMenu()

elif mode=='savetrakt' : traktit.traktIt('update', name)

elif mode=='restoretrakt' : traktit.traktIt('restore', name)

elif mode=='addontrakt' : traktit.traktIt('clearaddon', name)

elif mode=='cleartrakt' : traktit.clearSaved(name)

elif mode=='authtrakt' : traktit.activateTrakt(name); wiz.refresh()

elif mode=='updatetrakt' : traktit.autoUpdate('all')

elif mode=='importtrakt' : traktit.importlist(name); wiz.refresh()

elif mode=='realdebrid' : realMenu()

elif mode=='savedebrid' : debridit.debridIt('update', name)

elif mode=='restoredebrid' : debridit.debridIt('restore', name)

elif mode=='addondebrid' : debridit.debridIt('clearaddon', name)

elif mode=='cleardebrid' : debridit.clearSaved(name)

elif mode=='authdebrid' : debridit.activateDebrid(name); wiz.refresh()

elif mode=='updatedebrid' : debridit.autoUpdate('all')

elif mode=='importdebrid' : debridit.importlist(name); wiz.refresh()

elif mode=='alluc' : allucMenu()

elif mode=='savealluc' : allucit.allucIt('update', name)

elif mode=='restorealluc' : allucit.allucIt('restore', name)

elif mode=='addonalluc' : allucit.allucIt('clearaddon', name)

elif mode=='clearalluc' : allucit.clearSaved(name)

elif mode=='authalluc' : allucit.activateAlluc(name); wiz.refresh()

elif mode=='updatealluc' : allucit.autoUpdate('all')

elif mode=='importalluc' : allucit.importlist(name); wiz.refresh()

elif mode=='login' : loginMenu()

elif mode=='savelogin' : loginit.loginIt('update', name)

elif mode=='restorelogin' : loginit.loginIt('restore', name)

elif mode=='addonlogin' : loginit.loginIt('clearaddon', name)

elif mode=='clearlogin' : loginit.clearSaved(name)

elif mode=='authlogin' : loginit.activateLogin(name); wiz.refresh()

elif mode=='updatelogin' : loginit.autoUpdate('all')

elif mode=='importlogin' : loginit.importlist(name); wiz.refresh()

elif mode=='contact' : notify.contact(CONTACT)

elif mode=='settings' : wiz.openS(name); wiz.refresh()

elif mode=='forcetext' : wiz.forceText()

elif mode=='opensettings' : id = eval(url.upper()+'ID')[name]['plugin']; addonid = wiz.addonId(id); addonid.openSettings(); wiz.refresh()

elif mode=='developer' : developer()

elif mode=='converttext' : wiz.convertText()

elif mode=='createqr' : wiz.createQR()

elif mode=='testnotify' : testnotify()

elif mode=='testupdate' : testupdate()

elif mode=='testfirst' : testfirst()

elif mode=='testfirstrun' : testfirstRun()

###FTG MODS###

elif mode=='backup' : backup()

elif mode=='addon' : addon()

elif mode=='misc' : misc()

elif mode=='tweaks' : tweaks()

elif mode=='net' : net\_tools()

elif mode=='viewIP' : viewIP()

elif mode=='backup' : backup()

elif mode=='apk1' : apkMenu()

elif mode=='apkgame' : APKGAME(url)

elif mode=='select' : APKSELECT2(url)

elif mode=='grab' : APKGRAB(name,url)

elif mode=='rom' : romMenu(url)

elif mode=='apkscrape' : APK()

elif mode=='apkshow' : apkshowMenu(url)

elif mode=='apkkodi' : apkkodiMenu()

elif mode=='apkinstall' : apkInstaller(name, url,"None")

elif mode=='APPINSTALLER' : APPINSTALL(name,url,description)

elif mode=='ftgmod' : ftgmod()

elif mode=='viewpack' : viewpack()

elif mode=='addonpackwiz' : addonpackwiz()

elif mode=='FavsMenu' : FavsMenu()

elif mode=='savefav' : wiz.BACKUPFAV()

elif mode=='restorefav' : wiz.RESTOREFAV()

elif mode=='clearfav' : wiz.DELFAV()

elif mode=='apkfiles' : apkfiles()

elif mode=='retromenu' : retromenu()

elif mode=='emumenu' : emumenu()

elif mode=='rompackmenu' : rompackmenu()

elif mode=='UNZIPROM' : UNZIPROM()

elif mode=='ftgmod' : ftgmod()

elif mode=='GetList' : GetList(url)

elif mode=='autoadvanced' : notify.autoConfig2(); wiz.refresh()

elif mode=='autoconfig' : autoconfig()

elif mode=='sswap' : skinSwitch.popUPmenu()

### You have found my Lucky Charms !!

#MKDIRS()

window.connect(pyxbmct.ACTION\_NAV\_BACK, window.close)

window.doModal()

del window