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

# /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 uservar

from datetime import date, datetime, timedelta

from resources.libs import extract, downloader, notify, loginit, debridit, allucit, traktit, skinSwitch, uploadLog, wizard as wiz

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')

ADDONID = wiz.addonInfo(ADDON\_ID,'id')

DIALOG = xbmcgui.Dialog()

DP = xbmcgui.DialogProgress()

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

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

KODIHOME = xbmc.translatePath('special://xbmc/')

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

KODIADDONS = os.path.join(KODIHOME, 'addons')

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

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

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

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

TEXTCACHE = os.path.join(ADDONDATA, 'Cache')

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

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

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

SKIN = xbmc.getSkinDir()

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

BUILDNAME = wiz.getS('buildname')

DEFAULTSKIN = wiz.getS('defaultskin')

DEFAULTNAME = wiz.getS('defaultskinname')

DEFAULTIGNORE = wiz.getS('defaultskinignore')

BUILDVERSION = wiz.getS('buildversion')

BUILDLATEST = wiz.getS('latestversion')

BUILDCHECK = wiz.getS('lastbuildcheck')

DISABLEUPDATE = wiz.getS('disableupdate')

AUTOCLEANUP = wiz.getS('autoclean')

AUTOCACHE = wiz.getS('clearcache')

AUTOPACKAGES = wiz.getS('clearpackages')

AUTOTHUMBS = wiz.getS('clearthumbs')

AUTOFEQ = wiz.getS('autocleanfeq')

AUTONEXTRUN = wiz.getS('nextautocleanup')

TRAKTSAVE = wiz.getS('traktlastsave')

REALSAVE = wiz.getS('debridlastsave')

ALLUCSAVE = wiz.getS('alluclastsave')

LOGINSAVE = wiz.getS('loginlastsave')

KEEPTRAKT = wiz.getS('keeptrakt')

KEEPREAL = wiz.getS('keepdebrid')

KEEPALLUC = wiz.getS('keepalluc')

KEEPLOGIN = wiz.getS('keeplogin')

INSTALLED = wiz.getS('installed')

EXTRACT = wiz.getS('extract')

EXTERROR = wiz.getS('errors')

NOTIFY = wiz.getS('notify')

NOTEDISMISS = wiz.getS('notedismiss')

NOTEID = wiz.getS('noteid')

BACKUPLOCATION = ADDON.getSetting('path') if not ADDON.getSetting('path') == '' else HOME

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

NOTEID = 0 if NOTEID == "" else int(NOTEID)

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

TODAY = date.today()

TOMORROW = TODAY + timedelta(days=1)

TWODAYS = TODAY + timedelta(days=2)

THREEDAYS = TODAY + timedelta(days=3)

ONEWEEK = TODAY + timedelta(days=7)

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

EXCLUDES = uservar.EXCLUDES

BUILDFILE = uservar.BUILDFILE

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

WIZARDFILE = uservar.WIZARDFILE

AUTOINSTALL = uservar.AUTOINSTALL

REPOID = uservar.REPOID

REPOADDONXML = uservar.REPOADDONXML

REPOZIPURL = uservar.REPOZIPURL

COLOR1 = uservar.COLOR1

COLOR2 = uservar.COLOR2

WORKING = True if wiz.workingURL(BUILDFILE) == True else False

FAILED = False

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

#### Check Updates ######

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

def checkUpdate():

BUILDNAME = wiz.getS('buildname')

BUILDVERSION = wiz.getS('buildversion')

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

match = re.compile('name="%s".+?ersion="(.+?)".+?con="(.+?)".+?anart="(.+?)"' % BUILDNAME).findall(link)

if len(match) > 0:

version = match[0][0]

icon = match[0][1]

fanart = match[0][2]

wiz.setS('latestversion', version)

if version > BUILDVERSION:

if DISABLEUPDATE == 'false':

wiz.log("[Check Updates] [Installed Version: %s] [Current Version: %s] Opening Update Window" % (BUILDVERSION, version), xbmc.LOGNOTICE)

notify.updateWindow(BUILDNAME, BUILDVERSION, version, icon, fanart)

else: wiz.log("[Check Updates] [Installed Version: %s] [Current Version: %s] Update Window Disabled" % (BUILDVERSION, version), xbmc.LOGNOTICE)

else: wiz.log("[Check Updates] [Installed Version: %s] [Current Version: %s]" % (BUILDVERSION, version), xbmc.LOGNOTICE)

else: wiz.log("[Check Updates] ERROR: Unable to find build version in build text file", xbmc.LOGERROR)

def checkSkin():

wiz.log("[Build Check] Invalid Skin Check Start")

DEFAULTSKIN = wiz.getS('defaultskin')

DEFAULTNAME = wiz.getS('defaultskinname')

DEFAULTIGNORE = wiz.getS('defaultskinignore')

gotoskin = False

if not DEFAULTSKIN == '':

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

if DIALOG.yesno(ADDONTITLE, "[COLOR %s]It seems that the skin has been set back to [COLOR %s]%s[/COLOR]" % (COLOR2, COLOR1, SKIN[5:].title()), "Would you like to set the skin back to:[/COLOR]", '[COLOR %s]%s[/COLOR]' % (COLOR1, DEFAULTNAME)):

gotoskin = DEFAULTSKIN

gotoname = DEFAULTNAME

else: wiz.log("Skin was not reset", xbmc.LOGNOTICE); wiz.setS('defaultskinignore', 'true'); gotoskin = False

else: wiz.setS('defaultskin', ''); wiz.setS('defaultskinname', ''); DEFAULTSKIN = ''; DEFAULTNAME = ''

if DEFAULTSKIN == '':

skinname = []

skinlist = []

for folder in glob.glob(os.path.join(ADDONS, 'skin.\*/')):

xml = "%s/addon.xml" % folder

if os.path.exists(xml):

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

match = wiz.parseDOM(g, 'addon', ret='id')

match2 = wiz.parseDOM(g, 'addon', ret='name')

wiz.log("%s: %s" % (folder, str(match[0])), xbmc.LOGNOTICE)

if len(match) > 0: skinlist.append(str(match[0])); skinname.append(str(match2[0]))

else: wiz.log("ID not found for %s" % folder, xbmc.LOGNOTICE)

else: wiz.log("ID not found for %s" % folder, xbmc.LOGNOTICE)

if len(skinlist) > 0:

if len(skinlist) > 1:

if DIALOG.yesno(ADDONTITLE, "[COLOR %s]It seems that the skin has been set back to [COLOR %s]%s[/COLOR]" % (COLOR2, COLOR1, SKIN[5:].title()), "Would you like to view a list of avaliable skins?[/COLOR]"):

choice = DIALOG.select("Select skin to switch to!", skinname)

if choice == -1: wiz.log("Skin was not reset", xbmc.LOGNOTICE); wiz.setS('defaultskinignore', 'true')

else:

gotoskin = skinlist[choice]

gotoname = skinname[choice]

else: wiz.log("Skin was not reset", xbmc.LOGNOTICE); wiz.setS('defaultskinignore', 'true')

else:

if DIALOG.yesno(ADDONTITLE, "[COLOR %s]It seems that the skin has been set back to [COLOR %s]%s[/COLOR]" % (COLOR2, COLOR1, SKIN[5:].title()), "Would you like to set the skin back to:[/COLOR]", '[COLOR %s]%s[/COLOR]' % (COLOR1, skinname[0])):

gotoskin = skinlist[0]

gotoname = skinname[0]

else: wiz.log("Skin was not reset", xbmc.LOGNOTICE); wiz.setS('defaultskinignore', 'true')

else: wiz.log("No skins found in addons folder.", xbmc.LOGNOTICE); wiz.setS('defaultskinignore', 'true'); gotoskin = False

if gotoskin:

skinSwitch.swapSkins(gotoskin)

x = 0

xbmc.sleep(1000)

while not xbmc.getCondVisibility("Window.isVisible(yesnodialog)") and x < 150:

x += 1

xbmc.sleep(200)

if xbmc.getCondVisibility("Window.isVisible(yesnodialog)"):

wiz.ebi('SendClick(11)')

wiz.lookandFeelData('restore')

else: wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE),'[COLOR %s]Skin Swap Timed Out![/COLOR]' % COLOR2)

wiz.log("[Build Check] Invalid Skin Check End", xbmc.LOGNOTICE)

while xbmc.Player().isPlayingVideo():

xbmc.sleep(1000)

if KODIV >= 17:

NOW = datetime.now()

temp = wiz.getS('kodi17iscrap')

if not temp == '':

if temp > str(NOW - timedelta(minutes=2)):

wiz.log("Killing Start Up Script")

sys.exit()

wiz.log("%s" % (NOW))

wiz.setS('kodi17iscrap', str(NOW))

xbmc.sleep(1000)

if not wiz.getS('kodi17iscrap') == str(NOW):

wiz.log("Killing Start Up Script")

sys.exit()

else:

wiz.log("Continuing Start Up Script")

wiz.log("[Path Check] Started", xbmc.LOGNOTICE)

path = os.path.split(ADDONPATH)

if not ADDONID == path[1]: DIALOG.ok(ADDONTITLE, '[COLOR %s]Please make sure that the plugin folder is the same as the ADDON\_ID.[/COLOR]' % COLOR2, '[COLOR %s]Plugin ID:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, ADDONID), '[COLOR %s]Plugin Folder:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR2, COLOR1, path)); wiz.log("[Path Check] ADDON\_ID and plugin folder doesnt match. %s / %s " % (ADDONID, path))

else: wiz.log("[Path Check] Good!", xbmc.LOGNOTICE)

if KODIADDONS in ADDONPATH:

wiz.log("Copying path to addons dir", xbmc.LOGNOTICE)

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

newpath = xbmc.translatePath(os.path.join('special://home/addons/', ADDONID))

if os.path.exists(newpath):

wiz.log("Folder already exists, cleaning House", xbmc.LOGNOTICE)

wiz.cleanHouse(newpath)

wiz.removeFolder(newpath)

try:

wiz.copytree(ADDONPATH, newpath)

except Exception, e:

pass

wiz.forceUpdate(True)

try:

mybuilds = xbmc.translatePath(MYBUILDS)

if not os.path.exists(mybuilds): xbmcvfs.mkdirs(mybuilds)

except:

pass

wiz.log("[Auto Install Repo] Started", xbmc.LOGNOTICE)

if AUTOINSTALL == 'Yes' and not os.path.exists(os.path.join(ADDONS, REPOID)):

workingxml = wiz.workingURL(REPOADDONXML)

if workingxml == True:

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

if len(ver) > 0:

installzip = '%s-%s.zip' % (REPOID, ver[0])

workingrepo = wiz.workingURL(REPOZIPURL+installzip)

if workingrepo == True:

DP.create(ADDONTITLE,'Baixando Repo...','', 'Por favor, aguarde.. ')

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

lib=os.path.join(PACKAGES, installzip)

try: os.remove(lib)

except: pass

downloader.download(REPOZIPURL+installzip,lib, DP)

extract.all(lib, ADDONS, DP)

try:

f = open(os.path.join(ADDONS, REPOID, 'addon.xml'), mode='r'); g = f.read(); f.close()

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

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, name[0]), "[COLOR %s]Add-on updated[/COLOR]" % COLOR2, icon=os.path.join(ADDONS, REPOID, 'icon.png'))

except:

pass

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

DP.close()

xbmc.sleep(500)

wiz.forceUpdate(True)

wiz.log("[Auto Install Repo] Successfully Installed", xbmc.LOGNOTICE)

else:

wiz.LogNotify("[COLOR %s]Repo Install Error[/COLOR]" % COLOR1, "[COLOR %s]Invalid url for zip![/COLOR]" % COLOR2)

wiz.log("[Auto Install Repo] Was unable to create a working url for repository. %s" % workingrepo, xbmc.LOGERROR)

else:

wiz.log("Invalid URL for Repo Zip", xbmc.LOGERROR)

else:

wiz.LogNotify("[COLOR %s]Repo Install Error[/COLOR]" % COLOR1, "[COLOR %s]Invalid addon.xml file![/COLOR]" % COLOR2)

wiz.log("[Auto Install Repo] Unable to read the addon.xml file.", xbmc.LOGERROR)

elif not AUTOINSTALL == 'Yes': wiz.log("[Auto Install Repo] Not Enabled", xbmc.LOGNOTICE)

elif os.path.exists(os.path.join(ADDONS, REPOID)): wiz.log("[Auto Install Repo] Repository already installed")

wiz.log("[Auto Update Wizard] Started", xbmc.LOGNOTICE)

if AUTOUPDATE == 'Yes':

wiz.wizardUpdate('startup')

else: wiz.log("[Auto Update Wizard] Not Enabled", xbmc.LOGNOTICE)

wiz.log("[Notifications] Started", xbmc.LOGNOTICE)

if ENABLE == 'Yes':

if not NOTIFY == 'true':

url = wiz.workingURL(NOTIFICATION)

if url == True:

id, msg = wiz.splitNotify(NOTIFICATION)

if not id == False:

try:

id = int(id); NOTEID = int(NOTEID)

if id == NOTEID:

if NOTEDISMISS == 'false':

notify.notification(msg)

else: wiz.log("[Notifications] id[%s] Dismissed" % int(id), xbmc.LOGNOTICE)

elif id > NOTEID:

wiz.log("[Notifications] id: %s" % str(id), xbmc.LOGNOTICE)

wiz.setS('noteid', str(id))

wiz.setS('notedismiss', 'false')

notify.notification(msg=msg)

wiz.log("[Notifications] Complete", xbmc.LOGNOTICE)

except Exception, e:

wiz.log("Error on Notifications Window: %s" % str(e), xbmc.LOGERROR)

else: wiz.log("[Notifications] Text File not formated Correctly")

else: wiz.log("[Notifications] URL(%s): %s" % (NOTIFICATION, url), xbmc.LOGNOTICE)

else: wiz.log("[Notifications] Turned Off", xbmc.LOGNOTICE)

else: wiz.log("[Notifications] Not Enabled", xbmc.LOGNOTICE)

wiz.log("[Installed Check] Started", xbmc.LOGNOTICE)

if INSTALLED == 'true':

if KODIV >= 17:

wiz.kodi17Fix()

if SKIN in ['skin.confluence', 'skin.estuary']:

checkSkin()

FAILED = True

elif not EXTRACT == '100' and not BUILDNAME == "":

wiz.log("[Installed Check] Build was extracted %s/100 with [ERRORS: %s]" % (EXTRACT, EXTERROR), xbmc.LOGNOTICE)

yes=DIALOG.yesno(ADDONTITLE, '[COLOR %s]%s[/COLOR] [COLOR %s]was not installed correctly!' % (COLOR1, COLOR2, BUILDNAME), 'Installed: [COLOR %s]%s[/COLOR] / Error Count: [COLOR %s]%s[/COLOR]' % (COLOR1, EXTRACT, COLOR1, EXTERROR), 'Would you like to try again?[/COLOR]', nolabel='[B]No Thanks![/B]', yeslabel='[B]Retry Install[/B]')

wiz.clearS('build')

FAILED = True

if yes:

wiz.ebi("PlayMedia(plugin://%s/?mode=install&name=%s&url=fresh)" % (ADDON\_ID, urllib.quote\_plus(BUILDNAME)))

wiz.log("[Installed Check] Fresh Install Re-activated", xbmc.LOGNOTICE)

else: wiz.log("[Installed Check] Reinstall Ignored")

elif SKIN in ['skin.confluence', 'skin.estuary']:

wiz.log("[Installed Check] Incorrect skin: %s" % SKIN, xbmc.LOGNOTICE)

defaults = wiz.getS('defaultskin')

if not defaults == '':

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

skinSwitch.swapSkins(defaults)

x = 0

xbmc.sleep(1000)

while not xbmc.getCondVisibility("Window.isVisible(yesnodialog)") and x < 150:

x += 1

xbmc.sleep(200)

if xbmc.getCondVisibility("Window.isVisible(yesnodialog)"):

wiz.ebi('SendClick(11)')

wiz.lookandFeelData('restore')

if not wiz.currSkin() == defaults and not BUILDNAME == "":

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

FAILED = True

if gui == 'http://':

wiz.log("[Installed Check] Guifix was set to http://", xbmc.LOGNOTICE)

DIALOG.ok(ADDONTITLE, "[COLOR %s]It looks like the skin settings was not applied to the build." % COLOR2, "Sadly no gui fix was attatched to the build", "You will need to reinstall the build and make sure to do a force close[/COLOR]")

elif wiz.workingURL(gui):

yes=DIALOG.yesno(ADDONTITLE, '%s was not installed correctly!' % BUILDNAME, 'It looks like the skin settings was not applied to the build.', 'Would you like to apply the GuiFix?', nolabel='[B]No, Cancel[/B]', yeslabel='[B]Apply Fix[/B]')

if yes: wiz.ebi("PlayMedia(plugin://%s/?mode=install&name=%s&url=gui)" % (ADDON\_ID, urllib.quote\_plus(BUILDNAME))); wiz.log("[Installed Check] Guifix attempting to install")

else: wiz.log('[Installed Check] Guifix url working but cancelled: %s' % gui, xbmc.LOGNOTICE)

else:

DIALOG.ok(ADDONTITLE, "[COLOR %s]It looks like the skin settings was not applied to the build." % COLOR2, "Sadly no gui fix was attatched to the build", "You will need to reinstall the build and make sure to do a force close[/COLOR]")

wiz.log('[Installed Check] Guifix url not working: %s' % gui, xbmc.LOGNOTICE)

else:

wiz.log('[Installed Check] Install seems to be completed correctly', xbmc.LOGNOTICE)

if not wiz.getS('pvrclient') == "":

wiz.toggleAddon(wiz.getS('pvrclient'), 1)

wiz.ebi('StartPVRManager')

wiz.addonUpdates('reset')

if KEEPTRAKT == 'true': traktit.traktIt('restore', 'all'); wiz.log('[Installed Check] Restoring Trakt Data', xbmc.LOGNOTICE)

if KEEPREAL == 'true': debridit.debridIt('restore', 'all'); wiz.log('[Installed Check] Restoring Real Debrid Data', xbmc.LOGNOTICE)

if KEEPLOGIN == 'true': loginit.loginIt('restore', 'all'); wiz.log('[Installed Check] Restoring Login Data', xbmc.LOGNOTICE)

wiz.clearS('install')

else: wiz.log("[Installed Check] Not Enabled", xbmc.LOGNOTICE)

if FAILED == False:

wiz.log("[Build Check] Started", xbmc.LOGNOTICE)

if not WORKING:

wiz.log("[Build Check] Not a valid URL for Build File: %s" % BUILDFILE, xbmc.LOGNOTICE)

elif BUILDCHECK == '' and BUILDNAME == '':

wiz.log("[Build Check] First Run", xbmc.LOGNOTICE)

notify.firstRunSettings()

xbmc.sleep(500)

notify.firstRun()

xbmc.sleep(500)

wiz.setS('lastbuildcheck', str(NEXTCHECK))

elif not BUILDNAME == '':

wiz.log("[Build Check] Build Installed", xbmc.LOGNOTICE)

if SKIN in ['skin.confluence', 'skin.estuary'] and not DEFAULTIGNORE == 'true':

checkSkin()

wiz.log("[Build Check] Build Installed: Checking Updates", xbmc.LOGNOTICE)

wiz.setS('lastbuildcheck', str(NEXTCHECK))

checkUpdate()

elif BUILDCHECK <= str(TODAY):

wiz.log("[Build Check] Build Installed: Checking Updates", xbmc.LOGNOTICE)

wiz.setS('lastbuildcheck', str(NEXTCHECK))

checkUpdate()

else:

wiz.log("[Build Check] Build Installed: Next check isnt until: %s / TODAY is: %s" % (BUILDCHECK, str(TODAY)), xbmc.LOGNOTICE)

wiz.log("[Trakt Data] Started", xbmc.LOGNOTICE)

if KEEPTRAKT == 'true':

if TRAKTSAVE <= str(TODAY):

wiz.log("[Trakt Data] Saving all Data", xbmc.LOGNOTICE)

traktit.autoUpdate('all')

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

else:

wiz.log("[Trakt Data] Next Auto Save isnt until: %s / TODAY is: %s" % (TRAKTSAVE, str(TODAY)), xbmc.LOGNOTICE)

else: wiz.log("[Trakt Data] Not Enabled", xbmc.LOGNOTICE)

wiz.log("[Real Debrid Data] Started", xbmc.LOGNOTICE)

if KEEPREAL == 'true':

if REALSAVE <= str(TODAY):

wiz.log("[Real Debrid Data] Saving all Data", xbmc.LOGNOTICE)

debridit.autoUpdate('all')

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

else:

wiz.log("[Real Debrid Data] Next Auto Save isnt until: %s / TODAY is: %s" % (REALSAVE, str(TODAY)), xbmc.LOGNOTICE)

else: wiz.log("[Real Debrid Data] Not Enabled", xbmc.LOGNOTICE)

wiz.log("[Login Data] Started", xbmc.LOGNOTICE)

if KEEPLOGIN == 'true':

if LOGINSAVE <= str(TODAY):

wiz.log("[Login Data] Saving all Data", xbmc.LOGNOTICE)

loginit.autoUpdate('all')

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

else:

wiz.log("[Login Data] Next Auto Save isnt until: %s / TODAY is: %s" % (LOGINSAVE, str(TODAY)), xbmc.LOGNOTICE)

else: wiz.log("[Login Data] Not Enabled", xbmc.LOGNOTICE)

filesize = int(wiz.getS('filesize\_alert'))

filesize\_thumb = int(wiz.getS('filesizethumb\_alert'))

total\_size2 = 0

total\_size = 0

count = 0

total\_sizetext2 = "%.0f" % (total\_size2/1024000.0)

for dirpath, dirnames, filenames in os.walk(PACKAGES):

count = 0

for f in filenames:

count += 1

fp = os.path.join(dirpath, f)

total\_size += os.path.getsize(fp)

total\_sizetext = "%.0f" % (total\_size/1024000.0)

if int(total\_sizetext) > filesize:

wiz.clearPackagesStart(); wiz.refresh()

wiz.log("[Auto Cleaner] Package Cleaner Triggered", xbmc.LOGNOTICE)

for dirpath2, dirnames2, filenames2 in os.walk(THUMBS):

for f2 in filenames2:

fp2 = os.path.join(dirpath2, f2)

total\_size2 += os.path.getsize(fp2)

total\_sizetext2 = "%.0f" % (total\_size2/1024000.0)

if int(total\_sizetext2) > filesize\_thumb:

wiz.clearThumb(); wiz.refresh()

wiz.log("[Auto Cleaner] Thumbs Cleaner Triggered", xbmc.LOGNOTICE)

if wiz.getS('clearcache') == 'true':

wiz.clearCache(); wiz.refresh()

wiz.log("[Auto Cleaner] Thumbs Cleaner Triggered", xbmc.LOGNOTICE)

wiz.setS('kodi17iscrap', '')