'''

Copyright (C) 2014-2016 ddurdle

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.

You should have received a copy of the GNU General Public License

along with this program. If not, see <http://www.gnu.org/licenses/>.

'''

import sys

import cgi

# cloudservice - standard XBMC modules

import xbmcaddon

#http://stackoverflow.com/questions/1208916/decoding-html-entities-with-python/1208931#1208931

def \_callback(matches):

id = matches.group(1)

try:

return unichr(int(id))

except:

return id

def decode(data):

return re.sub("&#(\d+)(;|(?=\s))", \_callback, data).strip()

def getParameter(key,default=''):

try:

value = plugin\_queries[key]

if value == 'true':

return True

elif value == 'false':

return False

else:

return value

except:

return default

def getSetting(key,default=''):

try:

value = addon.getSetting(key)

if value == 'true':

return True

elif value == 'false':

return False

else:

return value

except:

return default

def parse\_query(query):

queries = cgi.parse\_qs(query)

q = {}

for key, value in queries.items():

q[key] = value[0]

q['mode'] = q.get('mode', 'main')

return q

plugin\_queries = None

try:

plugin\_queries = parse\_query(sys.argv[2][1:])

except:pass

# global variables

import addon\_parameters

addon = addon\_parameters.addon

#addon = xbmcaddon.Addon(id='plugin.video.gdrive-testing')

#addon = xbmcaddon.Addon(id='plugin.video.gdrive')

#

#

#

class settings:

# Settings

##

##

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

self.addon = addon

#self.integratedPlayer = self.getSetting('integrated\_player', False)

self.cc = getParameter('cc', self.getSetting('cc', True))

self.srt = getParameter('srt', self.getSetting('srt', True))

#self.srt\_folder = getParameter('srt\_folder', self.getSetting('srt\_folder', False))

self.strm = getParameter('strm', True) ## force to TRUE, set to false manually

self.username = getParameter('username', '')

self.setCacheParameters()

self.promptQuality = getParameter('promptquality', self.getSetting('prompt\_quality', True))

self.parseTV = self.getSetting('parse\_tv', True)

self.parseMusic = self.getSetting('parse\_music', True)

self.skipResume = self.getSetting('video\_skip', 0.98)

# self.cloudResume = self.getSetting('resumepoint', 0)

self.cloudResumePrompt = self.getSetting('resumeprompt', False)

# self.cloudSpreadsheet = self.getSetting('library\_filename', 'CLOUD\_DB')

self.tv\_watch = self.getSetting('tv\_db\_watch', False)

self.movie\_watch = self.getSetting('movie\_db\_watch', False)

self.seek = getParameter('seek', 0)

self.trace = getSetting('trace', False)

self.photoResolution = int(self.getSettingInt('photo\_resolution', 0))

if self.photoResolution == 0:

self.photoResolution = 1280

elif self.photoResolution == 1:

self.photoResolution = 1920

elif self.photoResolution == 2:

self.photoResolution = 3840

elif self.photoResolution == 3:

self.photoResolution = 7680

elif self.photoResolution == 4:

self.photoResolution = 15360

elif self.photoResolution == 5:

self.photoResolution = 720

else:

self.photoResolution = 99999

# self.thumbnailResolution = int(self.getSetting('thumb\_resolution', 0))

# if self.thumbnailResolution == 0:

# self.thumbnailResolution = 80

# elif self.thumbnailResolution == 1:

# self.thumbnailResolution = 120

# else:

# self.thumbnailResolution = 200

self.streamer = self.getSetting('streamer', True)

self.streamPort = int(self.getSetting('stream\_port', 8011))

self.encfsDownloadType = int(self.getSetting('encfs\_download\_type', 1))

def setVideoParameters(self):

self.resume = getParameter('resume', False)

self.playOriginal = getParameter('original', self.getSetting('never\_stream', False))

def setCacheParameters(self):

self.cache = getParameter('cache', False)

# self.download = self.getSetting('always\_cache', getParameter('download', False))

self.download = getParameter('download', getSetting('always\_cache', False))

self.play = getParameter('play', getSetting('always\_cache', False))

self.cachePath = self.getSetting('cache\_folder')

self.cacheSingle = self.getSetting('cache\_single')

self.cachePercent = self.getSetting('cache\_percent', 10)

self.cacheChunkSize = self.getSetting('chunk\_size', 32 \* 1024)

self.cacheContinue = self.getSetting('cache\_continue', False)

self.cacheSRT = self.getSetting('cache\_srt', False)

self.cacheThumbnails = self.getSetting('cache\_thumbnails', False)

if self.cache:

self.download = False

self.play = False

def setEncfsParameters(self):

self.encfsCacheSingle = self.getSetting('encfs\_cache\_single')

self.encfsCachePercent = self.getSetting('encfs\_cache\_percent', 10)

self.encfsCacheChunkSize = self.getSetting('encfs\_chunk\_size', 32 \* 1024)

self.encfsSource = self.getSetting('encfs\_source')

self.encfsTarget = self.getSetting('encfs\_target')

self.encfsContinue = self.getSetting('encfs\_continue', False)

self.encfsStream = self.getSetting('encfs\_stream', False)

self.encfsExp = self.getSetting('encfs\_exp', False)

self.encfsInode = int(self.getSetting('encfs\_inode', 0))

self.encfsLast = self.getSetting('encfs\_last', '')

def setCryptoParameters(self):

self.cryptoPassword = self.getSetting('crypto\_password')

self.cryptoSalt = self.getSetting('crypto\_salt')

def getParameter(self, key, default=''):

try:

value = plugin\_queries[key]

if value == 'true':

return True

elif value == 'false':

return False

else:

return value

except:

return default

def getSetting(self, key, default=''):

try:

value = self.addon.getSetting(key)

if value == 'true':

return True

elif value == 'false':

return False

else:

return value

except:

return default

def getSettingInt(self, key,default=0):

try:

return int(self.addon.getSetting(key))

except:

return default