#!/usr/bin/python

# Nukebox 2000 Admin Event Logging plugin class

# Authors: David Hyland and Diarmuid Ryan

# Role: Admin

# 21 March 2016

**from** time **import** sleep # import sleep module

**from** datetime **import** datetime # import datetime module

**class** **AdminEventLogging()**:

'''The AdminEventLogging class provides

functions to generate log files based

on Nukebox event codes'''

# ================================

# Event Type options

# > eventType = 'Object\_Deleted'

# > eventType = 'General\_Error'

# > eventType = 'Table\_Altered'

# ================================

**def** **\_\_init\_\_(**self, eventType): # Function to initialise an object

self.eventType = eventType # Event Type passed into AdminEventLogging() object

# IF statements to select actions based on Event Types

**if** eventType == 'Object\_Deleted': # Object\_Deleted event type

writeToFile = open('/home/administrator/Documents/Admin Log Files/Deleted\_Objects\_Log\_File.txt', 'a') # Append to Log file

writeToFile.write('Timestamp: ' + datetime.strftime(datetime.now(), '%Y-%m-%d %H:%M:%S') + ' Event Type: ' + eventType + '\n') # Append Timestamp and Event Type to Log file

writeToFile.close()

sleep(2) # sleep 2 seconds delay

**elif** eventType == 'General\_Error': # General\_Error event type

writeToFile = open('/home/administrator/Documents/Admin Log Files/General\_Error\_Log\_File.txt', 'a') # Append to Log file

writeToFile.write('Timestamp: ' + datetime.strftime(datetime.now(), '%Y-%m-%d %H:%M:%S') + ' Event Type: ' + eventType + '\n') # Append Timestamp and Event Type to Log file

writeToFile.close()

sleep(2) # sleep 2 seconds delay

**elif** eventType == 'Table\_Altered': # Table\_Altered event type

writeToFile = open('/home/administrator/Documents/Admin Log Files/Table\_Alteration\_Log\_File.txt', 'a') # Append to Log file

writeToFile.write('Timestamp: ' + datetime.strftime(datetime.now(), '%Y-%m-%d %H:%M:%S') + ' Event Type: ' + eventType + '\n') # Append Timestamp and Event Type to Log file

writeToFile.close()

sleep(2) # sleep 2 seconds delay

**else**:

**print**('Unknown Event Type')

# <><><><><><><><><><><><><><>><><><><><><><><><><><><>

**def** **\_\_str\_\_**(self): # \_STR\_ function for describing the object

**return** 'The AdminEventLogging object file output consists of a Timestamp and Event Type'

# <><><><><><><><><><><><><><>><><><><><><><><><><><><>

**print**('\n######################\n# CREATING LOG FILES #\n# PLEASE WAIT! #\n######################')

# Instantiate AdminEventLogging() objects

'''Create an instance of an AdminEventLogging

object and pass an Event Type to the

AdminEventLogging def \_init\_(self, eventType) method'''

eventInstance1 = AdminEventLogging('Object\_Deleted') #Instance, Class(Event Type)

eventInstance2 = AdminEventLogging('General\_Error') #Instance, Class(Event Type)

eventInstance3 = AdminEventLogging('Table\_Altered') #Instance, Class(Event Type)

eventInstance4 = AdminEventLogging('Object\_Deleted') #Instance, Class(Event Type)

eventInstance5 = AdminEventLogging('Object\_Deleted') #Instance, Class(Event Type)

eventInstance6 = AdminEventLogging('Object\_Deleted') #Instance, Class(Event Type)

eventInstance7 = AdminEventLogging('Table\_Altered') #Instance, Class(Event Type)

eventInstance8 = AdminEventLogging('Object\_Deleted') #Instance, Class(Event Type)

# <><><><><><><><><><><><><><>><><><><><><><><><><><><>

**print**('\n###################\n# LOG FILES READY #\n###################')