"""

This sample demonstrates an implementation of the Lex Code Hook Interface

in order to serve a sample bot which manages reservations for hotel rooms and car rentals.

Bot, Intent, and Slot models which are compatible with this sample can be found in the Lex Console

as part of the 'BookTrip' template.

For instructions on how to set up and test this bot, as well as additional samples,

visit the Lex Getting Started documentation http://docs.aws.amazon.com/lex/latest/dg/getting-started.html.

"""

import json

import datetime

import time

import os

import dateutil.parser

import logging

import pymysql

import sqlalchemy

from sqlalchemy import create\_engine

import pandas as pd

logger = logging.getLogger()

logger.setLevel(logging.DEBUG)

# --- Helpers that build all of the responses ---

def elicit\_slot(session\_attributes, intent\_name, slots, slot\_to\_elicit, message):

return {

'sessionAttributes': session\_attributes,

'dialogAction': {

'type': 'ElicitSlot',

'intentName': intent\_name,

'slots': slots,

'slotToElicit': slot\_to\_elicit,

'message': message

}

}

def confirm\_intent(session\_attributes, intent\_name, slots, message):

return {

'sessionAttributes': session\_attributes,

'dialogAction': {

'type': 'ConfirmIntent',

'intentName': intent\_name,

'slots': slots,

'message': message

}

}

def close(session\_attributes, fulfillment\_state, message):

response = {

'sessionAttributes': session\_attributes,

'dialogAction': {

'type': 'Close',

'fulfillmentState': fulfillment\_state,

'message': message

}

}

return response

def delegate(session\_attributes, slots):

return {

'sessionAttributes': session\_attributes,

'dialogAction': {

'type': 'Delegate',

'slots': slots

}

}

# --- Helper Functions ---

def safe\_int(n):

"""

Safely convert n value to int.

"""

if n is not None:

return int(n)

return n

def try\_ex(func):

"""

Call passed in function in try block. If KeyError is encountered return None.

This function is intended to be used to safely access dictionary.

Note that this function would have negative impact on performance.

"""

try:

return func()

except KeyError:

return None

def rec\_restaurant(intent\_request):

genre = intent\_request['currentIntent']['slots']['genre']

#user = 'ET8n-r7glWYqZhuR6GcdNw'

session\_attributes = intent\_request['sessionAttributes'] if intent\_request['sessionAttributes'] is not None else {}

# Create database connection

db\_uri = 'mysql+pymysql://admin:bLeY2vQFGD5XFR4@brobsoaw-mysql.cskt1xevcmmb.us-east-1.rds.amazonaws.com/yelp'

engine = create\_engine(db\_uri)

# connect to database engine

conn = engine.connect()

# Query MySQL database

query = f"SELECT DISTINCT name FROM yelp.gen\_rec\_restaurants WHERE categories LIKE '%%{genre}%%' AND stars\_y >= 4 LIMIT 5"

df = pd.read\_sql(query, engine)

# Format results as response

response = f"Here are your customized recommendations. Choice 1: {df['name'][0]}. " + f" Choice 2: {df['name'][1]}. " + f" Choice 3: {df['name'][2]}. " + f" Choice 4: {df['name'][3]}. \n " + f" Choice 5: {df['name'][4]}. \n "

#response = "test"

conn.close()

return close(

session\_attributes,

'Fulfilled',

{

'contentType': 'PlainText',

'content': response

}

)

# --- Intents ---

def dispatch(intent\_request):

"""

Called when the user specifies an intent for this bot.

"""

#logger.debug('dispatch genre={}, intentName={}'.format(intent\_request['genre'], intent\_request['currentIntent']['name']))

intent\_name = intent\_request['currentIntent']['name']

# Dispatch to your bot's intent handlers=

if intent\_name == 'GeneralRec':

return rec\_restaurant(intent\_request)

raise Exception('Intent with name ' + intent\_name + ' not supported')

# --- Main handler ---

def lambda\_handler(event, context):

"""

Route the incoming request based on intent.

The JSON body of the request is provided in the event slot.

"""

# By default, treat the user request as coming from the America/New\_York time zone.

os.environ['TZ'] = 'America/New\_York'

time.tzset()

logger.debug('event.bot.name={}'.format(event['bot']['name']))

return dispatch(event)