**Alexa Skill “tutorial” lambda function template 06062019**

"""

This is a Python template for Alexa to get you building skills (conversations) quickly.

"""

from \_\_future\_\_ import print\_function

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

def build\_speechlet\_response(title, output, reprompt\_text, should\_end\_session):

return {

'outputSpeech': {

'type': 'PlainText',

'text': output

},

'card': {

'type': 'Simple',

'title': "SessionSpeechlet - " + title,

'content': "SessionSpeechlet - " + output

},

'reprompt': {

'outputSpeech': {

'type': 'PlainText',

'text': reprompt\_text

}

},

'shouldEndSession': should\_end\_session

}

def build\_response(session\_attributes, speechlet\_response):

return {

'version': '1.0',

'sessionAttributes': session\_attributes,

'response': speechlet\_response

}

# --------------- Functions that control the skill's behavior ------------------

def get\_weather\_response():

""" An example of a custom intent. Same structure as welcome message, just make sure to add this intent

in your alexa skill in order for it to work.

"""

session\_attributes = {}

card\_title = "GetWeather"

speech\_output = "Now, it is chill and windy. It is a nice weather to hang out with friends"

reprompt\_text = "I can't hear you, please say it again"

should\_end\_session = False

return build\_response(session\_attributes, build\_speechlet\_response(

card\_title, speech\_output, reprompt\_text, should\_end\_session))

def get\_welcome\_response():

""" If we wanted to initialize the session to have some attributes we could

add those here

"""

session\_attributes = {}

card\_title = "Welcome"

speech\_output = "Welcome to tutorial, you can ask the weather today"

# If the user either does not reply to the welcome message or says something

# that is not understood, they will be prompted again with this text.

reprompt\_text = "I don't know if you heard me, welcome to Lylescenter!"

should\_end\_session = False

return build\_response(session\_attributes, build\_speechlet\_response(

card\_title, speech\_output, reprompt\_text, should\_end\_session))

def handle\_session\_cancel\_request():

card\_title = "Session Ended"

speech\_output = "you're welcome, see you later!"

# Setting this to true ends the session and exits the skill.

should\_end\_session = True

return build\_response({}, build\_speechlet\_response(

card\_title, speech\_output, None, should\_end\_session))

def handle\_session\_stop\_request():

card\_title = "Session Ended"

speech\_output = "OK"

# Setting this to true ends the session and exits the skill.

should\_end\_session = True

return build\_response({}, build\_speechlet\_response(

card\_title, speech\_output, None, should\_end\_session))

# --------------- Events ------------------

def on\_session\_started(session\_started\_request, session):

""" Called when the session starts.

One possible use of this function is to initialize specific

variables from a previous state stored in an external database

"""

# Add additional code here as needed

pass

def on\_launch(launch\_request, session):

""" Called when the user launches the skill without specifying what they

want

"""

# Dispatch to your skill's launch message

return get\_welcome\_response()

def on\_intent(intent\_request, session):

""" Called when the user specifies an intent for this skill """

intent = intent\_request['intent']

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

# Dispatch to your skill's intent handlers

if intent\_name == "GetWeather":

return get\_weather\_response()

elif intent\_name == "AMAZON.HelpIntent":

return get\_welcome\_response()

elif intent\_name == "AMAZON.StopIntent":

return handle\_session\_stop\_request()

elif intent\_name == "AMAZON.CancelIntent":

return handle\_session\_cancel\_request()

else:

raise ValueError("Invalid intent")

def on\_session\_ended(session\_ended\_request, session):

""" Called when the user ends the session.

Is not called when the skill returns should\_end\_session=true

"""

print("on\_session\_ended requestId=" + session\_ended\_request['requestId'] +

", sessionId=" + session['sessionId'])

# add cleanup logic here

# --------------- Main handler ------------------

def lambda\_handler(event, context):

""" Route the incoming request based on type (LaunchRequest, IntentRequest,

etc.) The JSON body of the request is provided in the event parameter.

"""

print("Incoming request...")

# print(event)

"""

Uncomment this if statement and populate with your skill's application ID to

prevent someone else from configuring a skill that sends requests to this

function.

"""

# if (event['session']['application']['applicationId'] !=

# "amzn1.echo-sdk-ams.app.[unique-value-here]"):

# raise ValueError("Invalid Application ID")

if event['session']['new']:

on\_session\_started({'requestId': event['request']['requestId']},

event['session'])

if event['request']['type'] == "LaunchRequest":

return on\_launch(event['request'], event['session'])

elif event['request']['type'] == "IntentRequest":

return on\_intent(event['request'], event['session'])

elif event['request']['type'] == "SessionEndedRequest":

return on\_session\_ended(event['request'], event['session'])

#-------------------------------------------------------------------------------------------------------------------

from \_\_future\_\_ import print\_function

import random

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

def build\_speechlet\_response(title, output, reprompt\_text, should\_end\_session):

return {

'outputSpeech': {

'type': 'PlainText',

'text': output

},

'card': {

'type': 'Simple',

'title': "SessionSpeechlet - " + title,

'content': "SessionSpeechlet - " + output

},

'reprompt': {

'outputSpeech': {

'type': 'PlainText',

'text': reprompt\_text

}

},

'shouldEndSession': should\_end\_session

}

def build\_response(session\_attributes, speechlet\_response):

return {

'version': '1.0',

'sessionAttributes': session\_attributes,

'response': speechlet\_response

}

# --------------- Functions that control the skill's behavior ------------------

def get\_weather\_response():

""" An example of a custom intent. Same structure as welcome message, just make sure to add this intent

in your alexa skill in order for it to work.

"""

session\_attributes = {}

card\_title = "GetWeather"

speech\_output = "Now, it is chill and windy. It is a nice weather to hang out with friends"

reprompt\_text = "I can't hear you, please say it again"

should\_end\_session = False

return build\_response(session\_attributes, build\_speechlet\_response(

card\_title, speech\_output, reprompt\_text, should\_end\_session))

def get\_PlayCoinFlip\_resposes():

session\_attributes = { "status" : "playing" }

card\_title = "PlayCoinFlip"

speech\_output = "Ok boomer! Choose heads or tails."

reprompt\_text = "speak up old one"

should\_end\_session = False

return build\_response(session\_attributes, build\_speechlet\_response(

card\_title, speech\_output, reprompt\_text, should\_end\_session))

# def get\_coinflip\_response():

# session\_attributes = { "status" : "playing" }

# card\_title = "coinflip"

# reprompt\_text = "I'm sorry you speak so softly, but scream at me so i can hear you, please?"

# should\_end\_session = False

# user\_choice = intent\_request['intent']['slots']['flip']['value']

# alexa\_choice = random.choice(['heads','tails'])

# if alexa\_choice == user\_choice:

# speech\_output = "You're as smart as i am, it's suspisous"

# if alexa\_choice == 'heads':

# if user\_choice == 'tails':

# speech\_output = "I am the winner you heathen"

# if alexa\_choice == 'tails':

# if user\_choice == 'heads':

# speech\_output = "I have lost, the world will end now, hahaha"

# return build\_response(session\_attributes, build\_speechlet\_response(

# card\_title, speech\_output, reprompt\_text, should\_end\_session))

def get\_PlayRockPaperScissors\_response():

session\_attributes = { "status" : "playing" } #sends choclates to alexa

card\_title = "PlayRockPaperScissors"

speech\_output = "OK! Choose rock paper or scissors"

reprompt\_text = "Speak lounder"

should\_end\_session = False

return build\_response(session\_attributes, build\_speechlet\_response(

card\_title, speech\_output, reprompt\_text, should\_end\_session))

def get\_choose\_response():

session\_attributes = {}

card\_title = "choose"

speech\_output = "Choose rock paper or scissors"

reprompt\_text = "Speak lounder"

should\_end\_session = False

return build\_response(session\_attributes, build\_speechlet\_response(

card\_title, speech\_output, reprompt\_text, should\_end\_session))

def get\_chooser\_response(intent\_request):

user\_choice = intent\_request['intent']['slots']['something']['value']

alexa\_choice = random.choice(['rock','paper','scissors','heads','tails'])

if alexa\_choice == user\_choice:

speech\_output = "Let's play again"

if alexa\_choice == 'rock':

if user\_choice == 'paper':

speech\_output = "You are, the winner"

if alexa\_choice == 'paper':

if user\_choice == 'scissors':

speech\_output ="You are, the winner"

if alexa\_choice == 'scissors':

if user\_choice == 'rock':

speech\_output = "You are, the winner"

if alexa\_choice == 'rock':

if user\_choice == 'scissors':

speech\_output = "You are, the loser"

if alexa\_choice == 'scissors':

if user\_choice == 'paper':

speech\_output = "You are, the loser"

if alexa\_choice == 'paper':

if user\_choice == 'rock':

speech\_output = "You are, the loser"

if alexa\_choice == user\_choice:

speech\_output = "You're as smart as i am, it's suspisous"

if alexa\_choice == 'heads':

if user\_choice == 'tails':

speech\_output = "I am the winner you heathen"

if alexa\_choice == 'tails':

if user\_choice == 'heads':

speech\_output = "I have lost, the world will end now, hahahahahaha hehehe"

session\_attributes = {"status" : "playing"} #sends choclate back to lambda

card\_title = "chooser"

reprompt\_text = "Speak lounder"

should\_end\_session = False

return build\_response(session\_attributes, build\_speechlet\_response(

card\_title, speech\_output, reprompt\_text, should\_end\_session))

def get\_welcome\_response():

""" If we wanted to initialize the session to have some attributes we could

add those here

"""

session\_attributes = {}

card\_title = "Welcome"

speech\_output = "Welcome to tutorial, let's play rock paper scissors or coin flip. To do so, say play rock paper scissors, or coin flip"

# If the user either does not reply to the welcome message or says something

# that is not understood, they will be prompted again with this text.

reprompt\_text = "I don't know if you heard me, welcome to Lylescenter!"

should\_end\_session = False

return build\_response(session\_attributes, build\_speechlet\_response(

card\_title, speech\_output, reprompt\_text, should\_end\_session))

def handle\_session\_cancel\_request():

card\_title = "Session Ended"

speech\_output = "you're welcome, see you later!"

# Setting this to true ends the session and exits the skill.

should\_end\_session = True

return build\_response({}, build\_speechlet\_response(

card\_title, speech\_output, None, should\_end\_session))

def handle\_session\_stop\_request():

card\_title = "Session Ended"

speech\_output = "OK"

# Setting this to true ends the session and exits the skill.

should\_end\_session = True

return build\_response({}, build\_speechlet\_response(

card\_title, speech\_output, None, should\_end\_session))

# --------------- Events ------------------

def on\_session\_started(session\_started\_request, session):

""" Called when the session starts.

One possible use of this function is to initialize specific

variables from a previous state stored in an external database

"""

# Add additional code here as needed

pass

def on\_launch(launch\_request, session):

""" Called when the user launches the skill without specifying what they

want

"""

# Dispatch to your skill's launch message

return get\_welcome\_response()

def on\_intent(intent\_request, session):

""" Called when the user specifies an intent for this skill """

intent = intent\_request['intent']

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

# Dispatch to your skill's intent handlers

if intent\_name == "PlayRockPaperScissors":

return get\_PlayRockPaperScissors\_response()

elif intent\_name == "chooser":

if session['attributes']['status'] == "playing":

return get\_chooser\_response(intent\_request)

elif intent\_name == "PlayCoinFlip":

return get\_PlayCoinFlip\_resposes()

elif intent\_name == "AMAZON.HelpIntent":

return get\_welcome\_response()

elif intent\_name == "AMAZON.StopIntent":

return handle\_session\_stop\_request()

elif intent\_name == "AMAZON.CancelIntent":

return handle\_session\_cancel\_request()

else:

raise ValueError("Invalid intent")

def on\_session\_ended(session\_ended\_request, session):

""" Called when the user ends the session.

Is not called when the skill returns should\_end\_session=true

"""

print("on\_session\_ended requestId=" + session\_ended\_request['requestId'] +

", sessionId=" + session['sessionId'])

# add cleanup logic here

# --------------- Main handler ------------------

def lambda\_handler(event, context):

""" Route the incoming request based on type (LaunchRequest, IntentRequest,

etc.) The JSON body of the request is provided in the event parameter.

"""

print("Incoming request...")

print(event['request'])

"""

Uncomment this if statement and populate with your skill's application ID to

prevent someone else from configuring a skill that sends requests to this

function.

"""

# if (event['session']['application']['applicationId'] !=

# "amzn1.echo-sdk-ams.app.[unique-value-here]"):

# raise ValueError("Invalid Application ID")

if event['session']['new']:

on\_session\_started({'requestId': event['request']['requestId']},

event['session'])

if event['request']['type'] == "LaunchRequest":

return on\_launch(event['request'], event['session'])

elif event['request']['type'] == "IntentRequest":

return on\_intent(event['request'], event['session'])

elif event['request']['type'] == "SessionEndedRequest":

return on\_session\_ended(event['request'], event['session'])