**Alexa skill session and persistent attributes in python**

You can save temporary data in a session using session attributes, but they are lost If you close the session. To save data across sessions, you use persistent attributes.

For example, you might want to keep a record of how many times someone has used your skill and congratulate them when they reach a milestone (say every 10 visits)

To do this you need some storage and we will use an Alexa Hosted Skill which provides some DynamoDB database storage.

**Instructions:**

Start a new python Alexa Hosted skill:

Create Skill > name it (my welcome skill) > choose Custom and Alexa-hosted Python and Create skill. Then Start from Scratch and Continue with template

When it’s been created click Code and follow these instructions:

1. Edit requirements.txt and **add the following db adapter, then save it**

ask-sdk-dynamodb-persistence-adapter==1.15.0

1. Edit requirements.txt and **add this code to the top of lamba\_function.py**

import os

import boto3

from ask\_sdk\_dynamodb.adapter import DynamoDbAdapter

1. **Add (say before the class Launch..) the code to initialise the dDB**

ddb\_region = os.environ.get('DYNAMODB\_PERSISTENCE\_REGION')

ddb\_table\_name = os.environ.get('DYNAMODB\_PERSISTENCE\_TABLE\_NAME')

ddb\_resource = boto3.resource('dynamodb', region\_name=ddb\_region)

dynamodb\_adapter = DynamoDbAdapter(table\_name=ddb\_table\_name, create\_table=False, dynamodb\_resource=ddb\_resource)

from ask\_sdk\_core.skill\_builder import CustomSkillBuilder

from ask\_sdk\_dynamodb.adapter import DynamoDbAdapter

1. **Add the CustomSkillBuilder** to your skill builder request handlers, and the bottom of the code. (after sb = SkillBuilder())

sb = CustomSkillBuilder(persistence\_adapter = dynamodb\_adapter)

We’ll use persistence attributes to load and save data. See

<https://developer.amazon.com/en-US/docs/alexa/hosted-skills/alexa-hosted-skills-session-persistence.html#persistence-python>

We need to load and save the persistent attributes:

To load (read) them we use:

persistent\_attributes = handler\_input.attributes\_manager.persistent\_attributes

and to save them we use:

handler\_input.attributes\_manager.session\_attributes = persistent\_attributes

handler\_input.attributes\_manager.save\_persistent\_attributes()

We will need to set them up the first time the program runs, for example:

attr = handler\_input.attributes\_manager.persistent\_attributes

if not attr:

# First time - set attributes, e.g.:

attr['counter'] = 0

attr['state'] = 'STARTED’

If you want to read and save session attributes as well use:

# read

session\_attributes = handler\_input.attributes\_manager.session\_attributes

session\_ attributes ['state'] = "STARTED"

session\_ attributes ["counter"] = 1

# save

handler\_input.attributes\_manager.persistent\_attributes = session\_attributes

handler\_input.attributes\_manager.save\_persistent\_attributes()

Let’s see how this works:

**Change Launch request** to:

class LaunchRequestHandler(AbstractRequestHandler):

"""Handler for Skill Launch."""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_request\_type("LaunchRequest")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

persistent\_attributes = handler\_input.attributes\_manager.persistent\_attributes

# to delete use:

# handler\_input.attributes\_manager.delete\_persistent\_attributes()

session\_attributes = handler\_input.attributes\_manager.session\_attributes

if not persistent\_attributes:

persistent\_attributes['visit\_counter'] = 1

session\_attributes['number\_of\_hellos'] = 0

speak\_output = "Hi. This is visit number " + str(persistent\_attributes['visit\_counter']) + ". You haven't said Hello yet"

persistent\_attributes['visit\_counter'] = persistent\_attributes['visit\_counter'] + 1

handler\_input.attributes\_manager.persistent\_attributes = persistent\_attributes

handler\_input.attributes\_manager.save\_persistent\_attributes()

handler\_input.attributes\_manager.session\_attributes = session\_attributes

return (

handler\_input.response\_builder

.speak(speak\_output)

.ask(speak\_output)

.response

)

And the HelloWorldIntent to:

class HelloWorldIntentHandler(AbstractRequestHandler):

"""Handler for Hello World Intent."""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("HelloWorldIntent")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

session\_attributes = handler\_input.attributes\_manager.session\_attributes

session\_attributes['number\_of\_hellos'] += 1

speak\_output = "Hi. This is hello number " + str(session\_attributes['number\_of\_hellos'])

handler\_input.attributes\_manager.session\_attributes = session\_attributes

return (

handler\_input.response\_builder

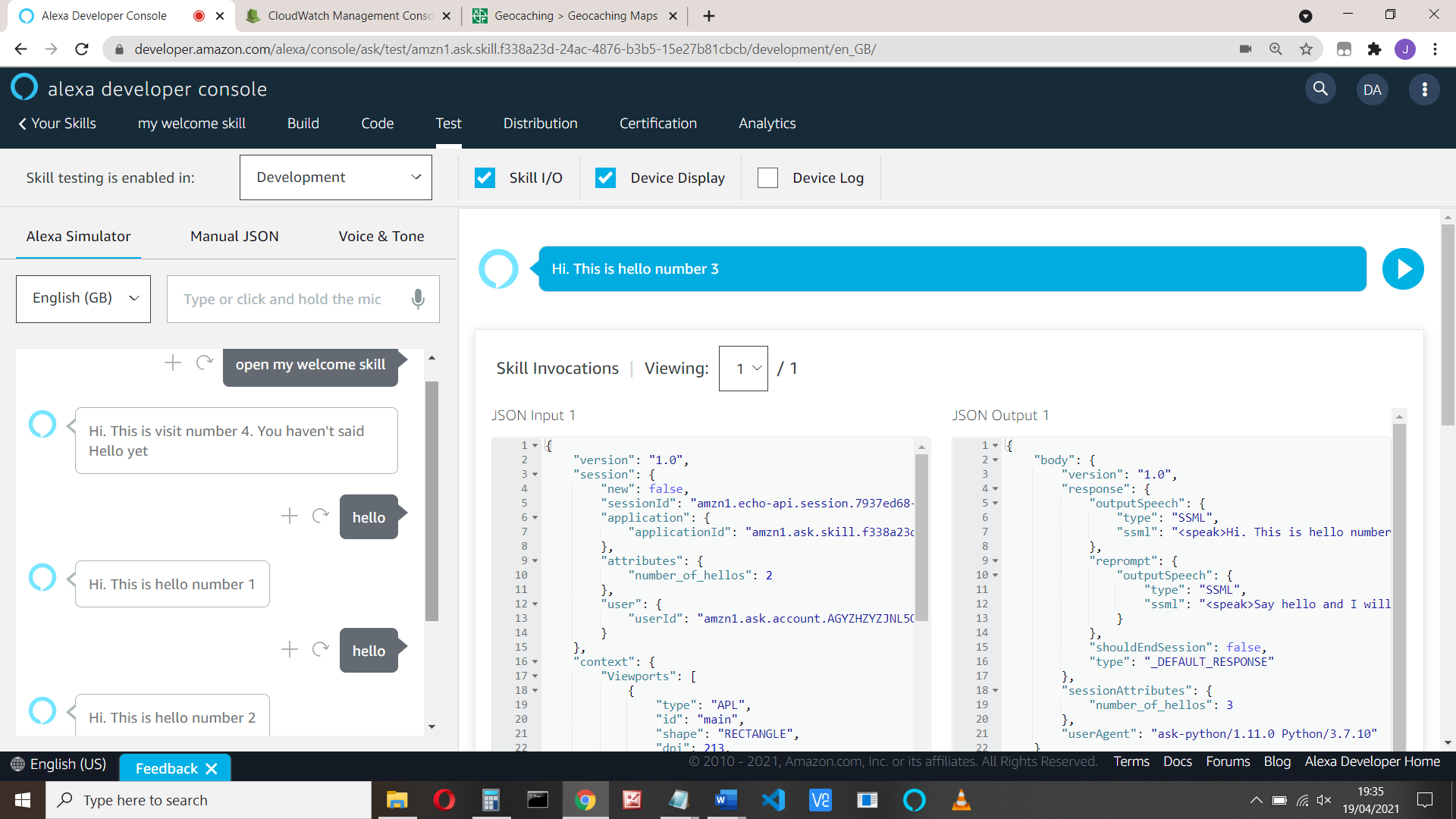
.speak(speak\_output)

.ask("Say hello and I will count them for you")

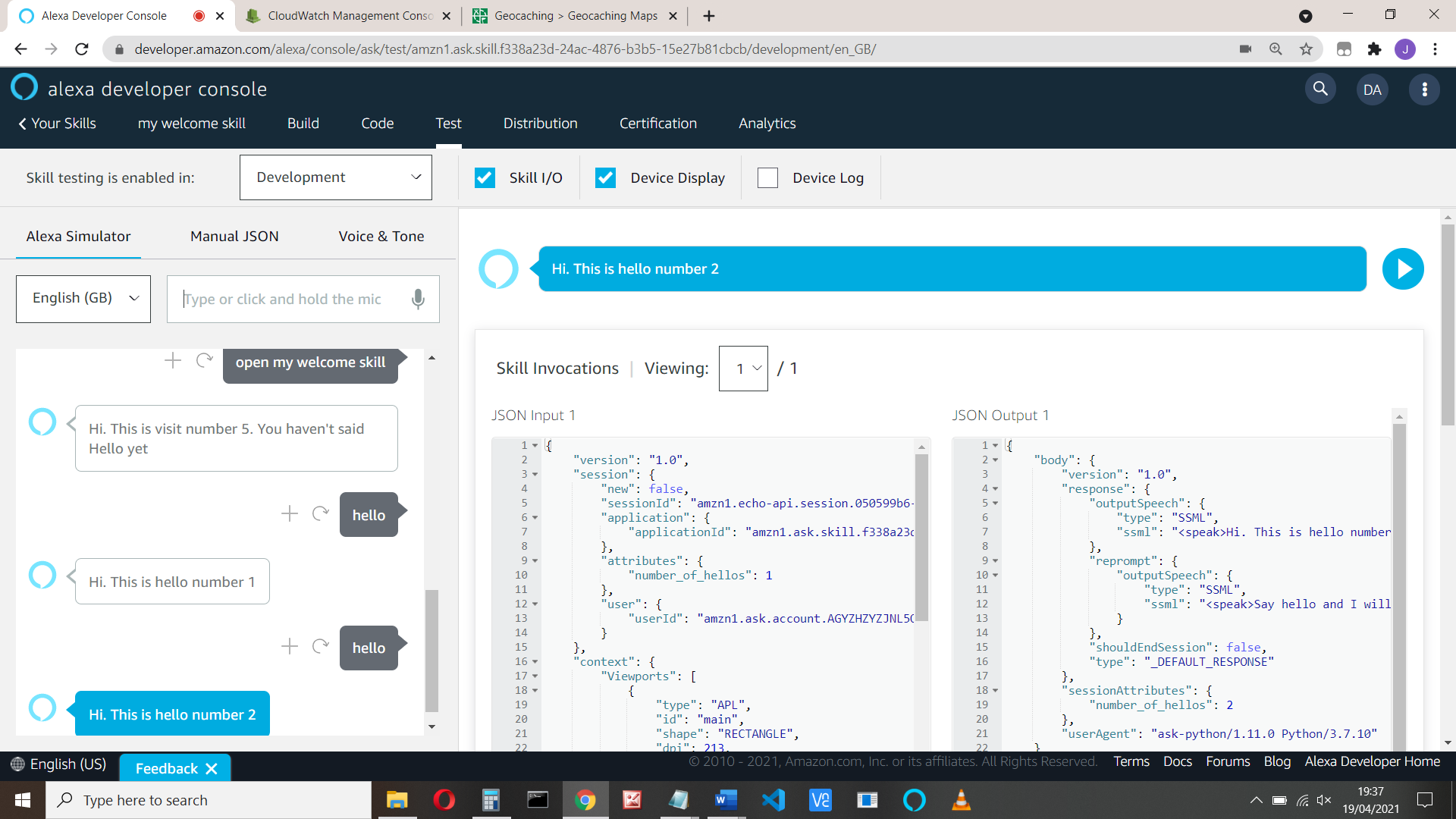
.response

)

**Run the skill a few times.** The hello message should increase each time you say hello and the visit count for each new visit.



Then later…



That’s it