**Alexa C# coding using Visual Studio**

**Part 5 – using slots**

Slots are how data is passed from your VUI to your program. As an example, we will get Alexa / your program to ask for your name.

The VUI might go like this:

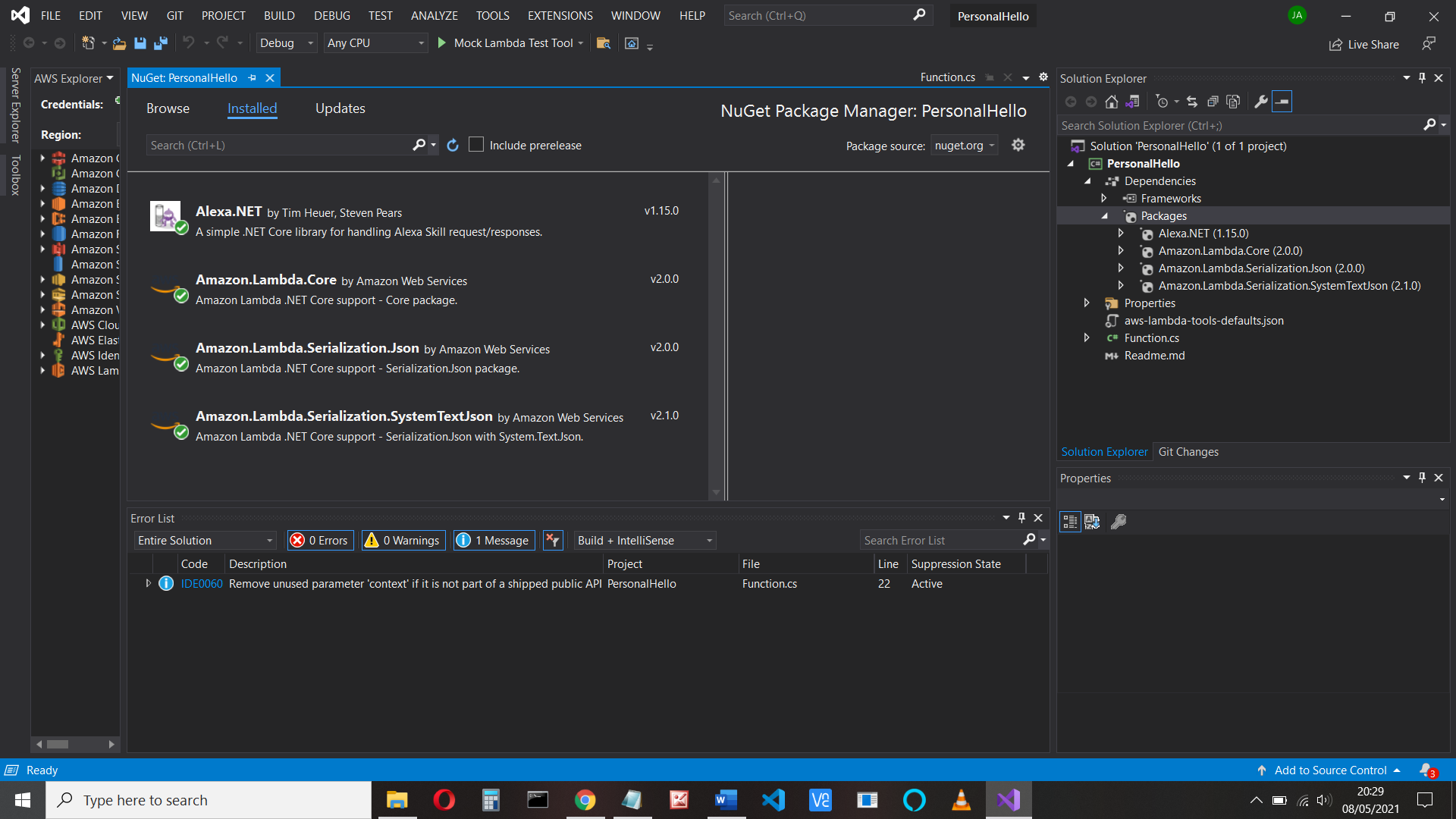
Alexa: Welcome, please tell me your name

User: My name is John (John will be passed from Alexa to your program in a slot)

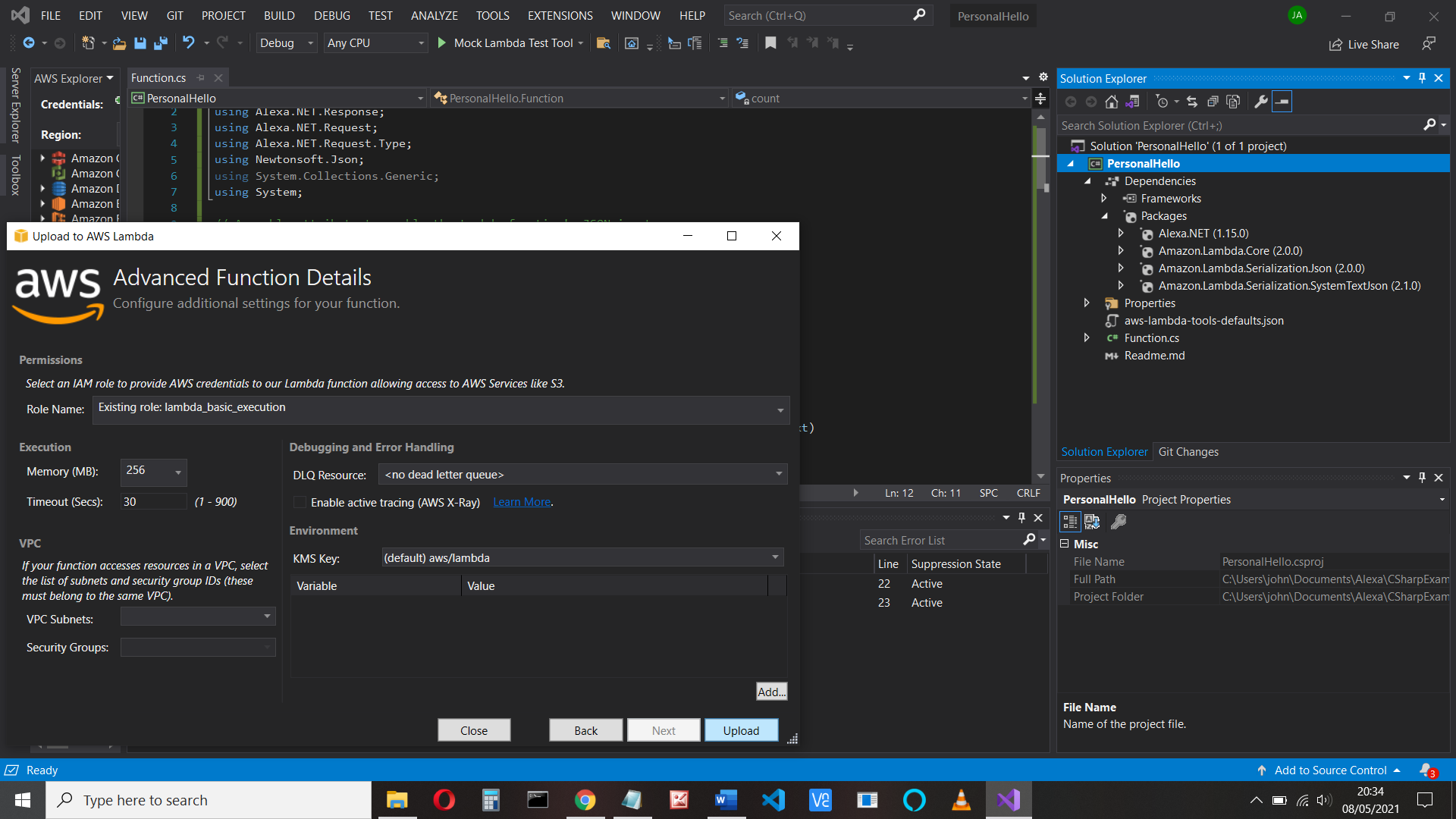
Alexa: Hello John (Your program picks John from the Alexa request and responds)

1. Create a new **AWS Lambda Project (.NET Core – C#)**
2. Configure your project – give it a name and location, choose ‘Empty function’ then Finish. Add your credentials if you haven’t done so.
3. Replace the function.cs code with the code that’s at the end. **Change the namespace if you’ve called it anything but PersonalHello**
4. Check for updates (Amazon.Lambda.Core is 2.0.0)
5. Add the libraries

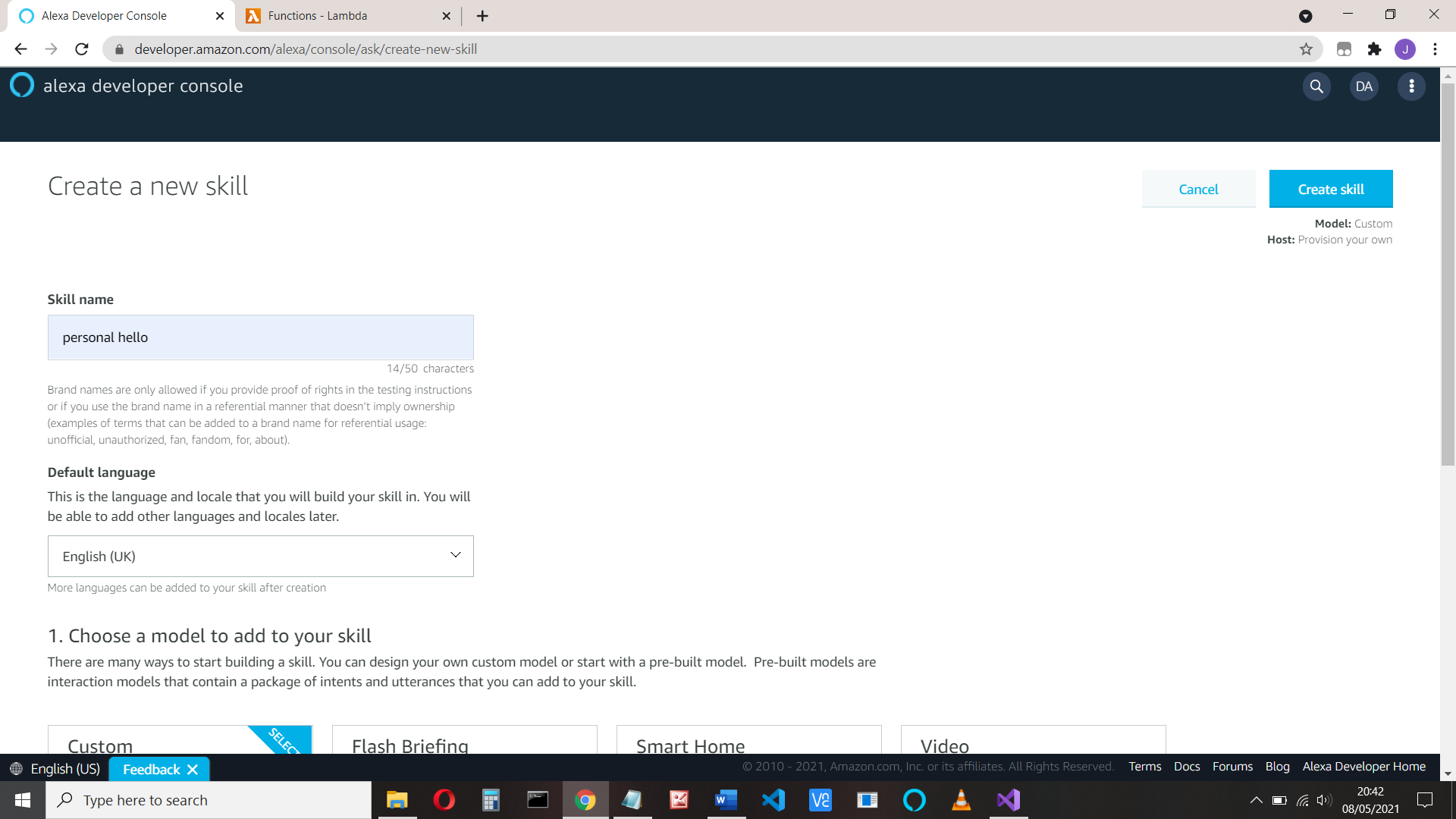
* Alexa.NET
* Amazon.Lambda.Serialization.Json
* You may see others installed, e.g. Amazon.Lambda.Serialization.System.TextJson



1. Save all, then upload. You may have to enter lambda\_basic\_execution as the Role



1. Test with an Alexa Start Session
2. Open developer.com in your browser, log in.
3. In developer, create a new skill, call it personal hello, use Custom skill and Provision Your Own code

and the Start from Scratch

1. Go to endpoints and copy the skill endpoint (amzn1.ask.skill.61212345 .. )
2. Open aws.amazon.com in another browser tab, log in and find your function.

Select Configuration, Add trigger, and add Alexa Skills Kit. Add the Skill endpoint.

Copy the lambda function ARN (arn:aws:lambda:us-east-1:55…2:function:PersonalHello)

1. Return to the developer skill and paste the ARN into to Default Region in the Endpoint. Now the skill is linked to the function and vice versa
2. Check the invocation is what you expect
3. Click Interaction model > Intents and delete the HelloWorldIntent

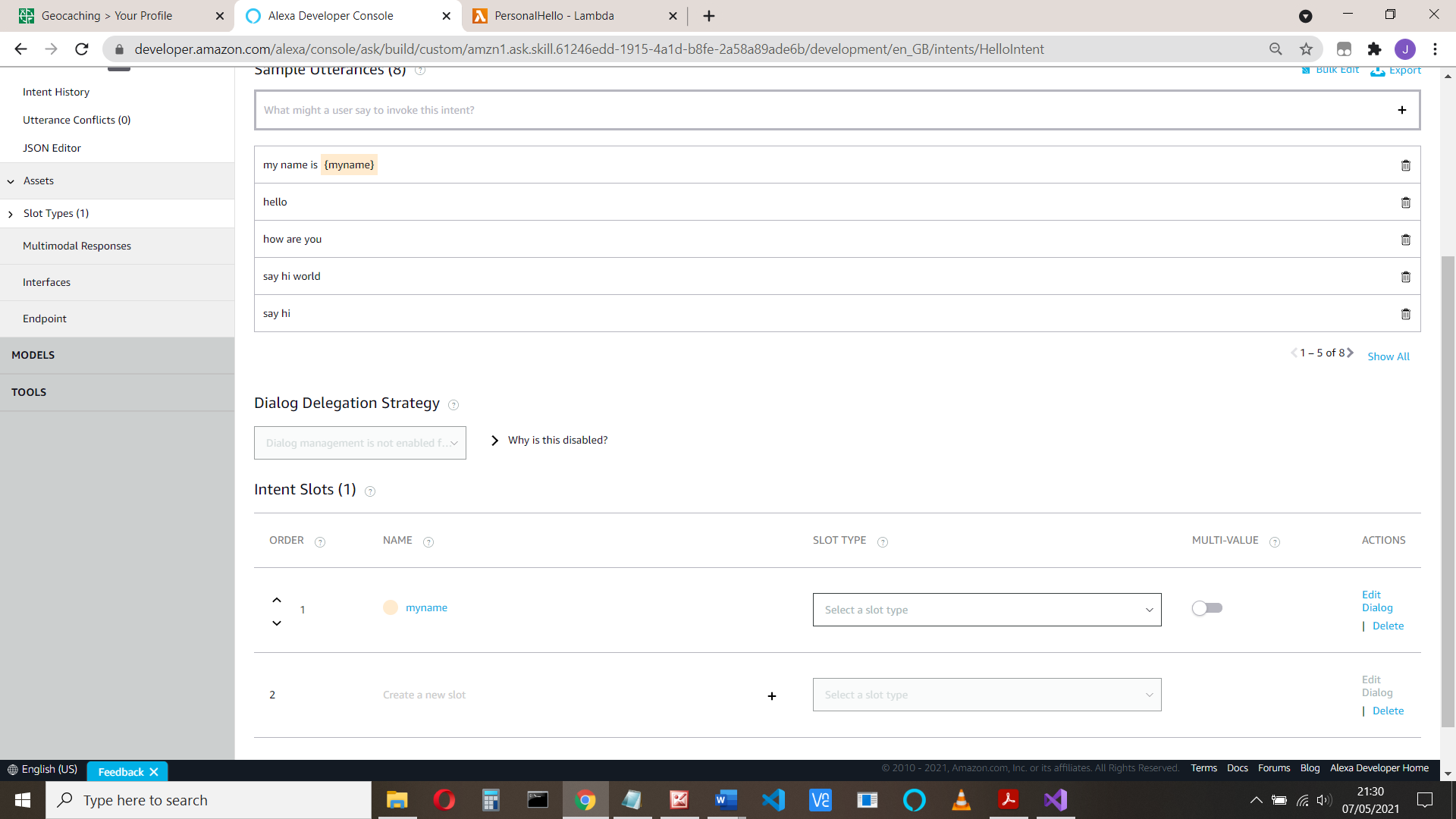
Now add our HelloIntent intent (Click +Add Intent) and utterances (what the user might say).

Our user might say their name, or just Hello or Howdy.

Add the utterances, including those which might use a name. To add a name, use the slot ‘myname’ in curly brackets, e,g, *My name is {myname}*

Type in My name is {myname} and click the + sign to add the intent

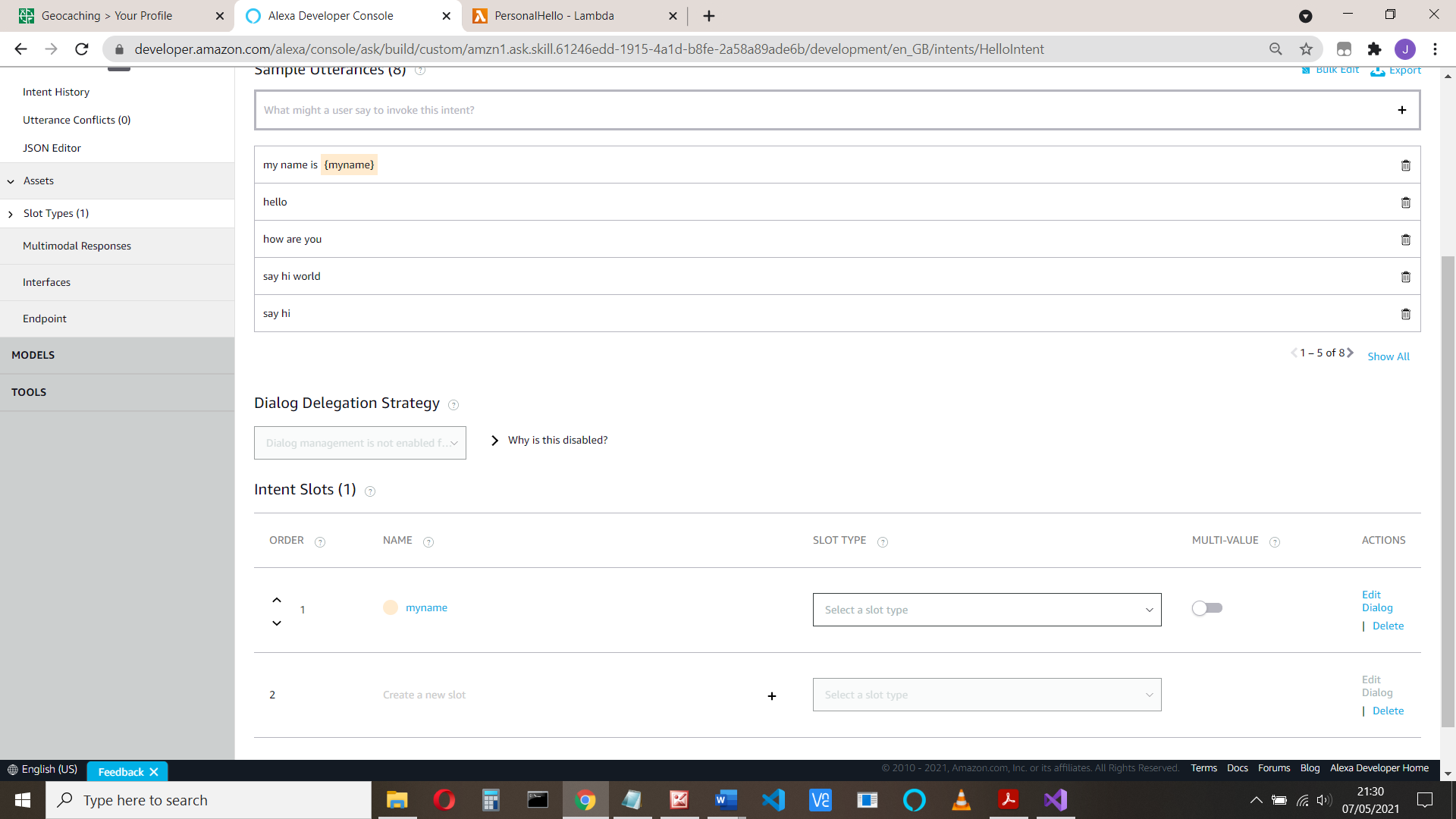
Now scroll down to see the slot



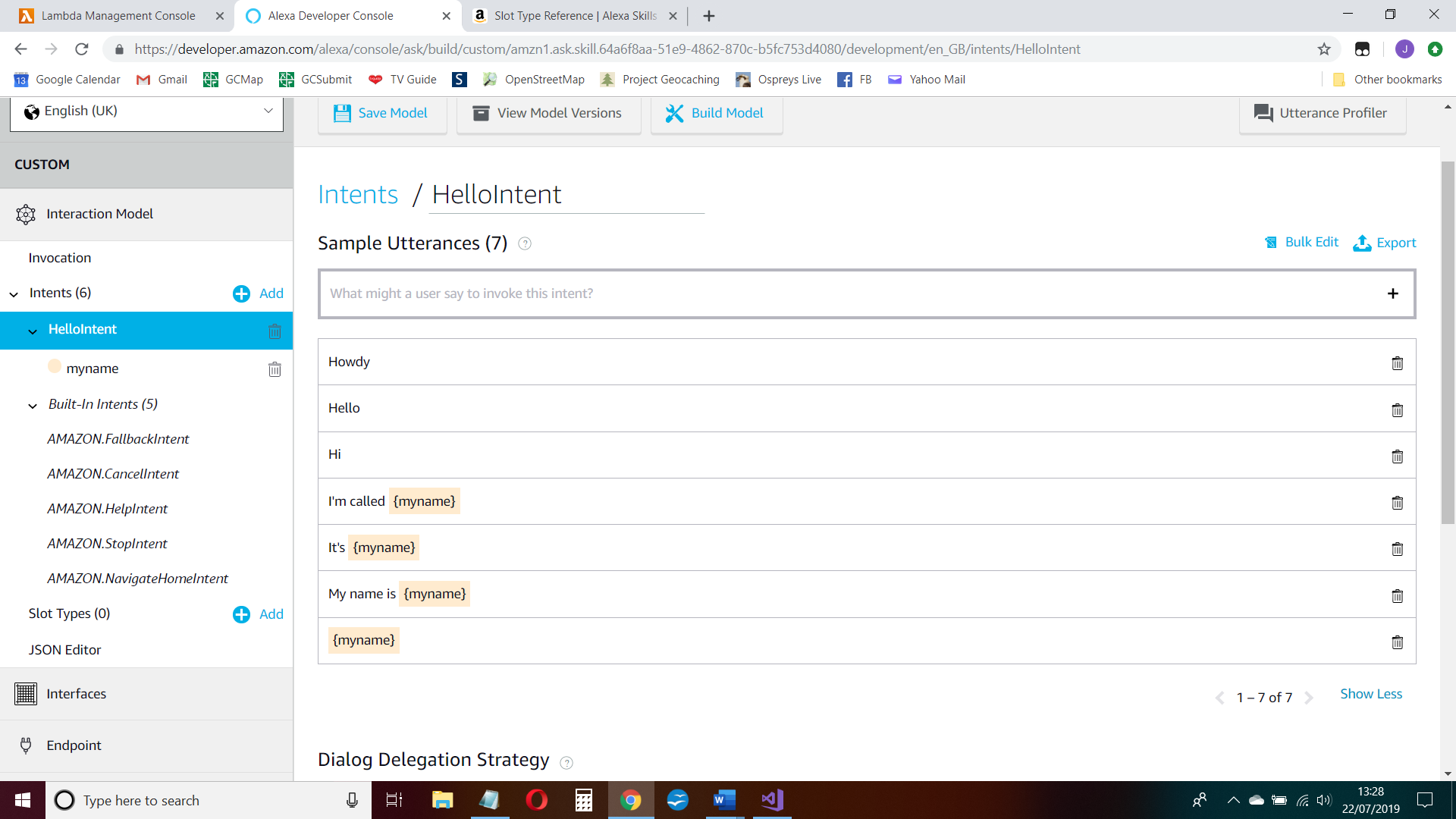
We need to define the myname slot type (what sort of data does it hold?).

Amazon has plenty of built-in slot types (e.g. date, times, numbers, phone numbers, etc.) and also AMAZON.FirstName which we can use here.

Scroll down and select AMAZON.FirstName



Add any other utterances you can think of, e.g. They are: I’m called {myname}, I’m {myname}, My name is {myname}, It’s {myname}, and {myname}



Note that we can tick boxes to make sure the slot is filled (Alexa will check this) or require confirmation. Leave these alone for now.

The relevant part of the JSON is:

{

"name": "HelloIntent",

"slots": [

{

"name": "myname",

"type": "AMAZON.FirstName"

}

],

"samples": [

"{myname}",

"It's {myname}",

"I'm {myname}",

"I'm called {myname}",

"my name is {myname}",

"hello",

"how are you",

"say hi world",

"say hi",

"hi",

"say hello world",

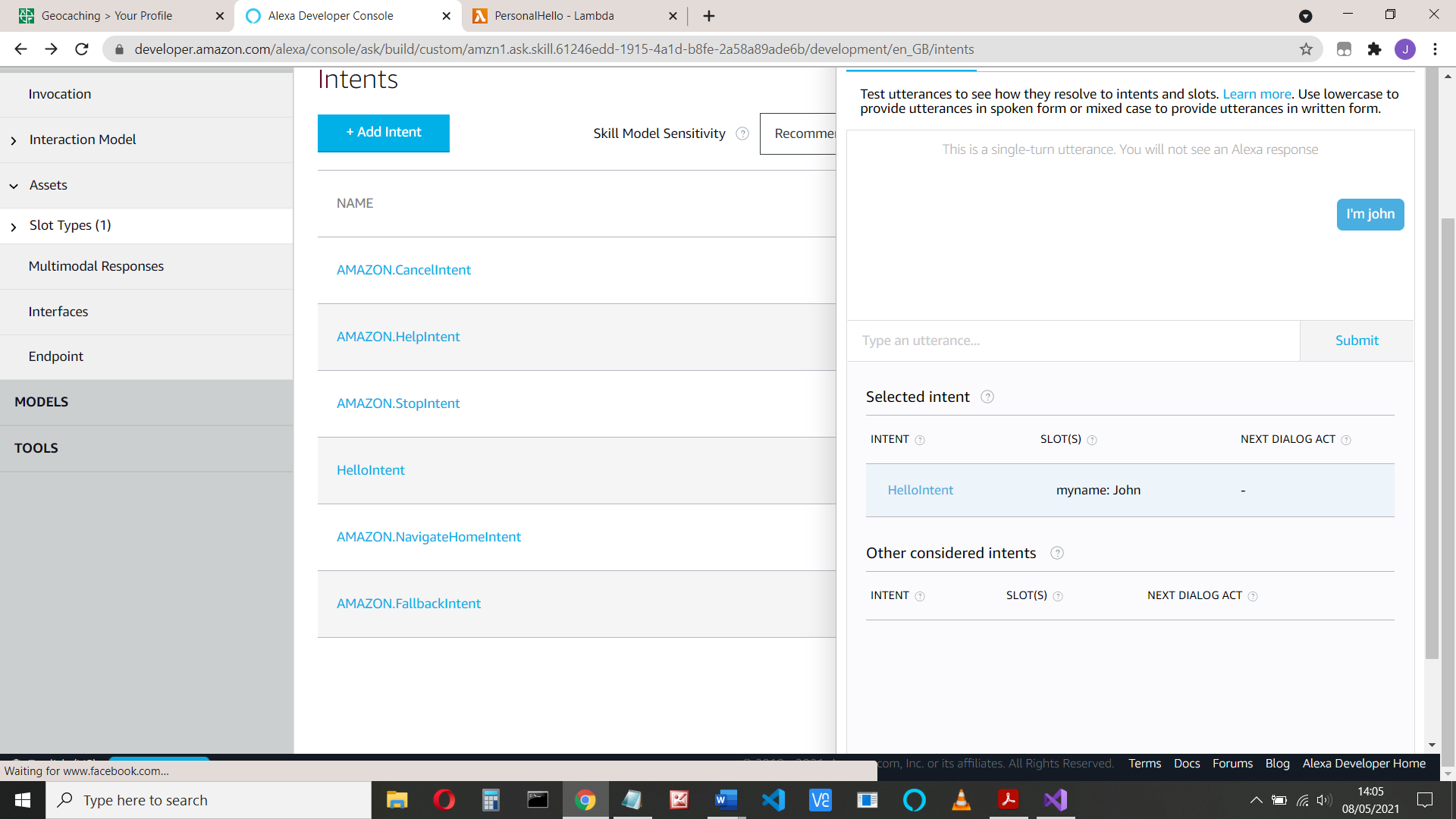
"say hello"

]

}

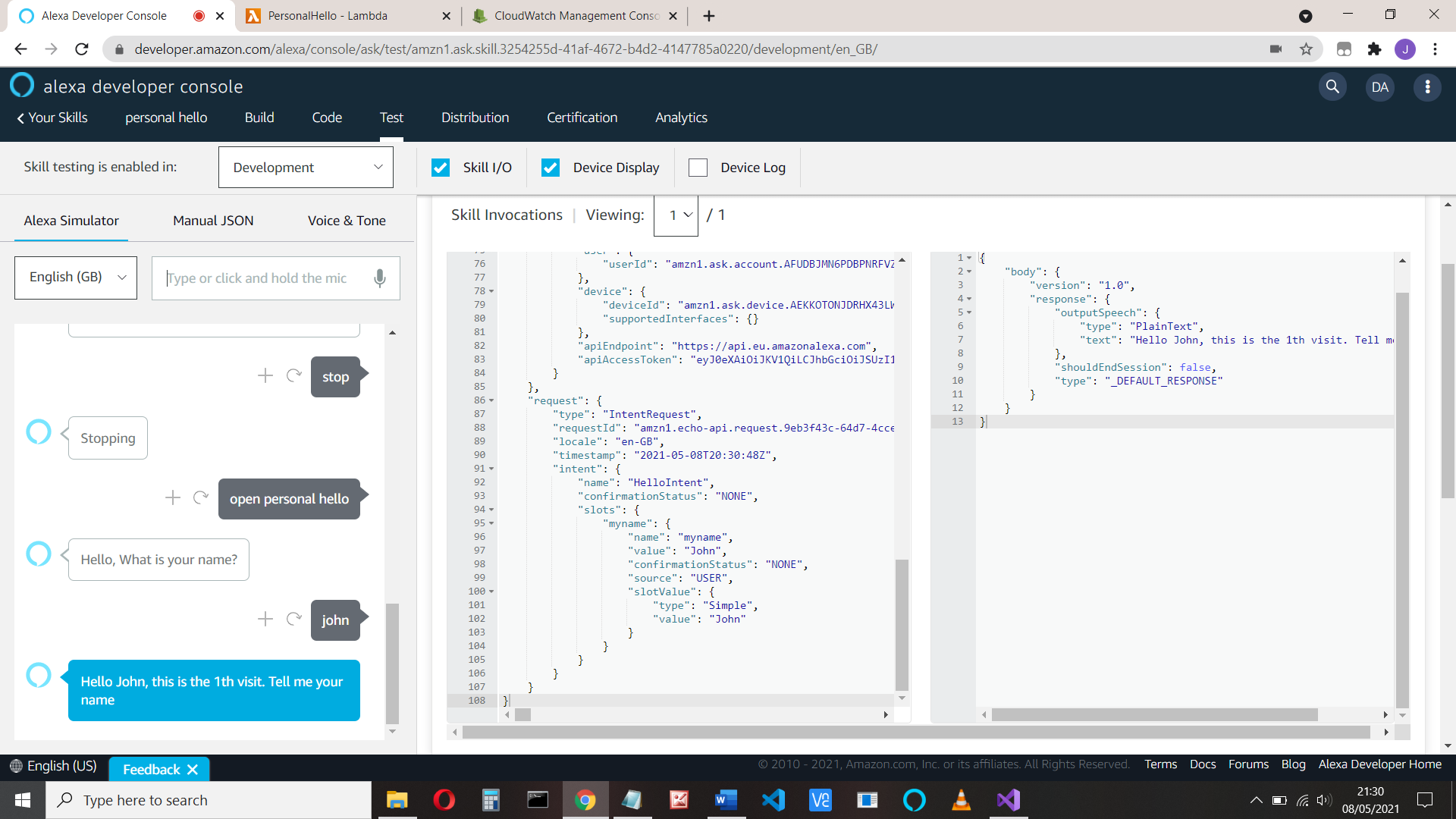
Save and build the model

You can test the intent. Click **Evaluate Model** and type ‘I’m John’ into the Utterance Profiler and see the HelloIntent will be used.



Now we need to see how myname is passed to our program.

1. Click the Test tab to test the skill, Select Development from the skill testing drop-down box
2. Run the program from the Test area (type in: open personal hello, then your name when asked) and look at the JSON input code request that is sent to our program.



You can see that ‘myname’ is a **slot** which has the name ‘myname’ and the value ‘John’. More accurately, it is:

intentRequest.Intent.Slots["myname"].Value

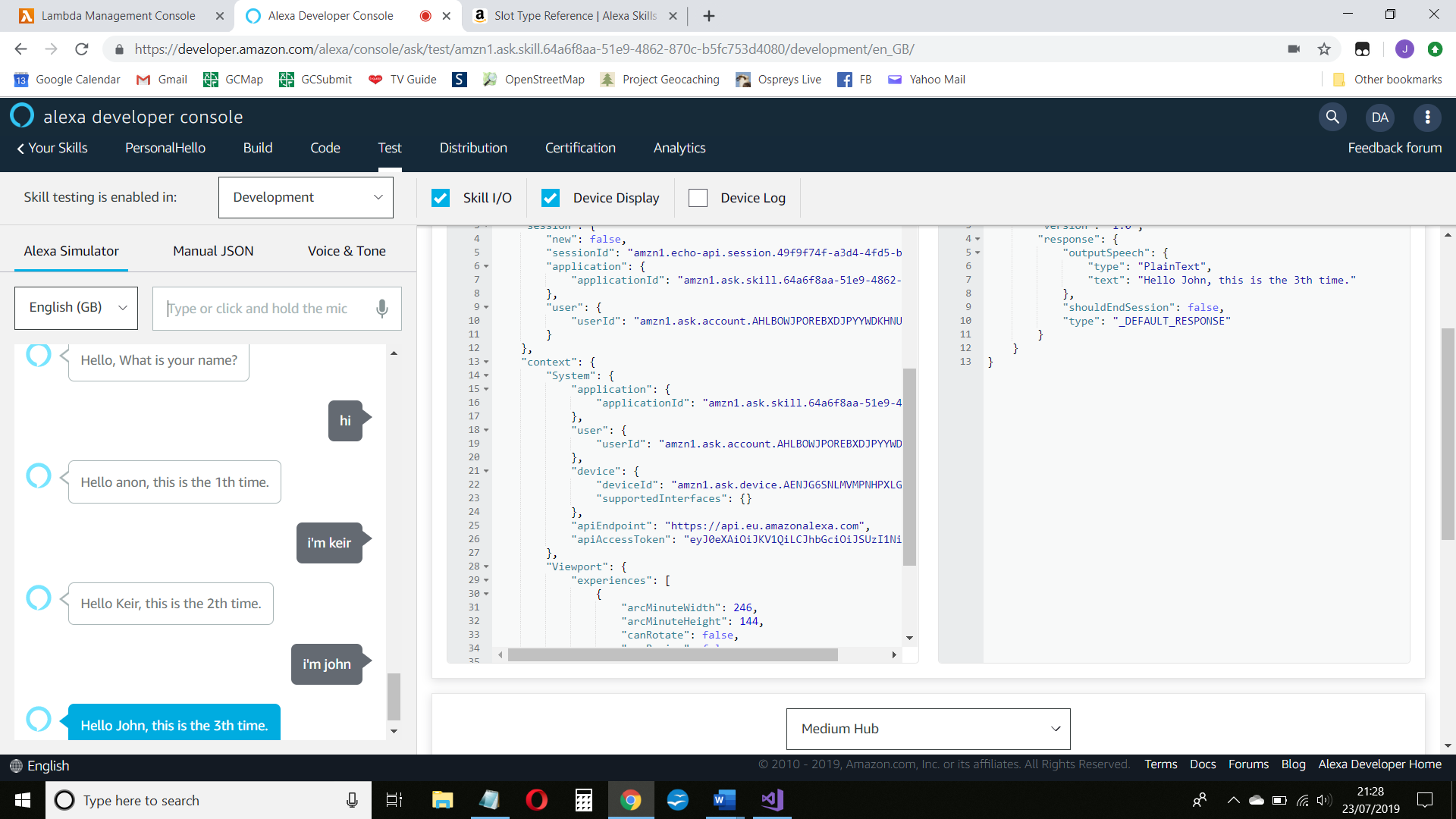
And that’s how we access the ‘myname’ value in our program. We use:

userName = intentRequest.Intent.Slots["myname"].Value;

Another example might be:

int playersGuess = Convert.ToInt32(intentRequest.Intent.Slots["number"].Value);

Testing it shows the following:



The HelloIntent code runs when the user replies. The code is:

case "HelloIntent":

count++;

{

if (userName == null) // we don't know their name, call them anon

{

userName = "anon";

}

}

else // myname has a value, use that

{

userName = intentRequest.Intent.Slots["myname"].Value;

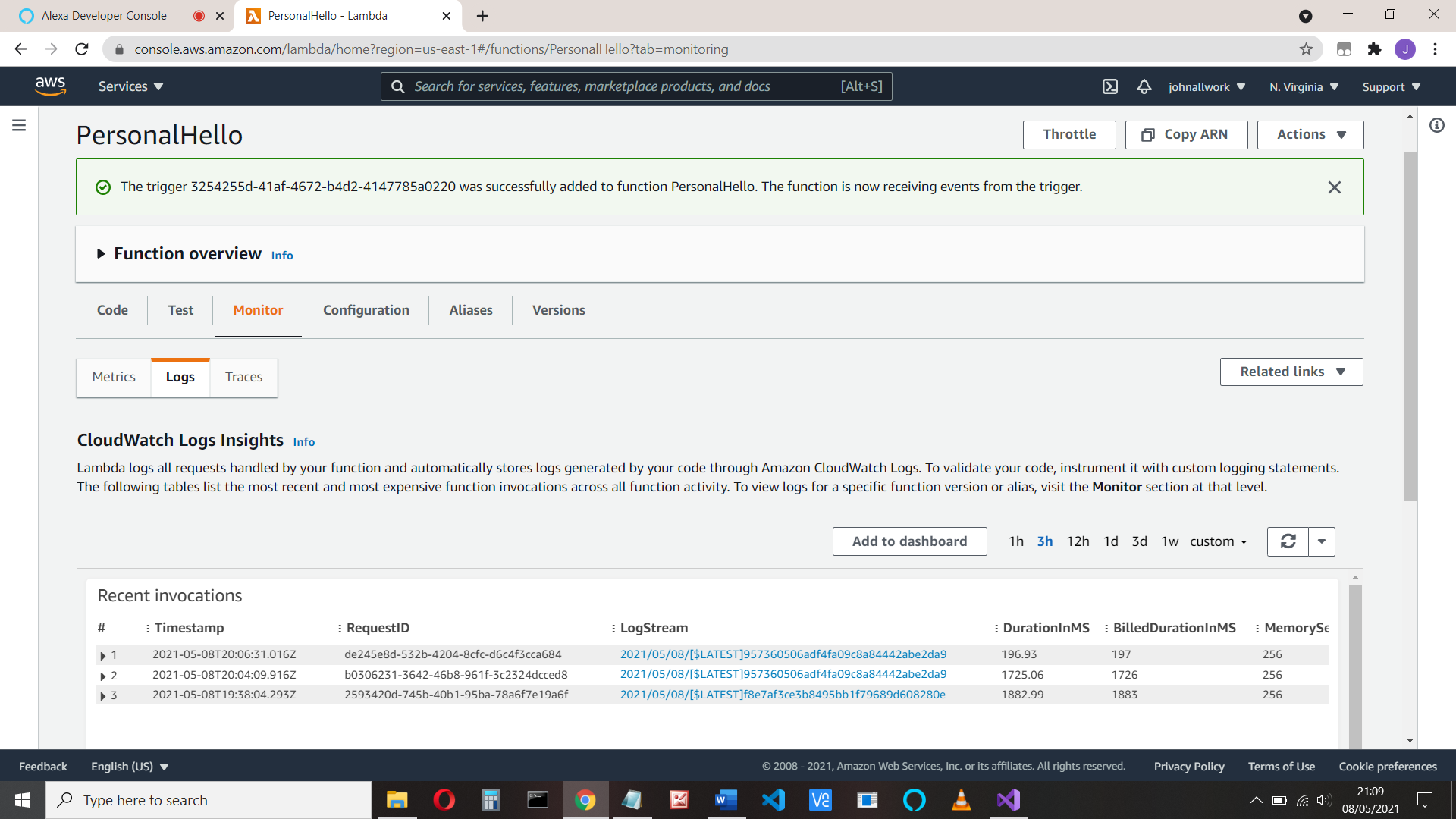
}

innerResponse = new PlainTextOutputSpeech();

(innerResponse as PlainTextOutputSpeech).Text = $"Hello {userName}, this is the {count}th visit. Tell me your name";

break;

1. If you need to debug your code, or see the logs from your program, go to your function in aws lambda and select Monitor > Logs:



Then select the log stream

You can see the logs which include the request and response, from our code:

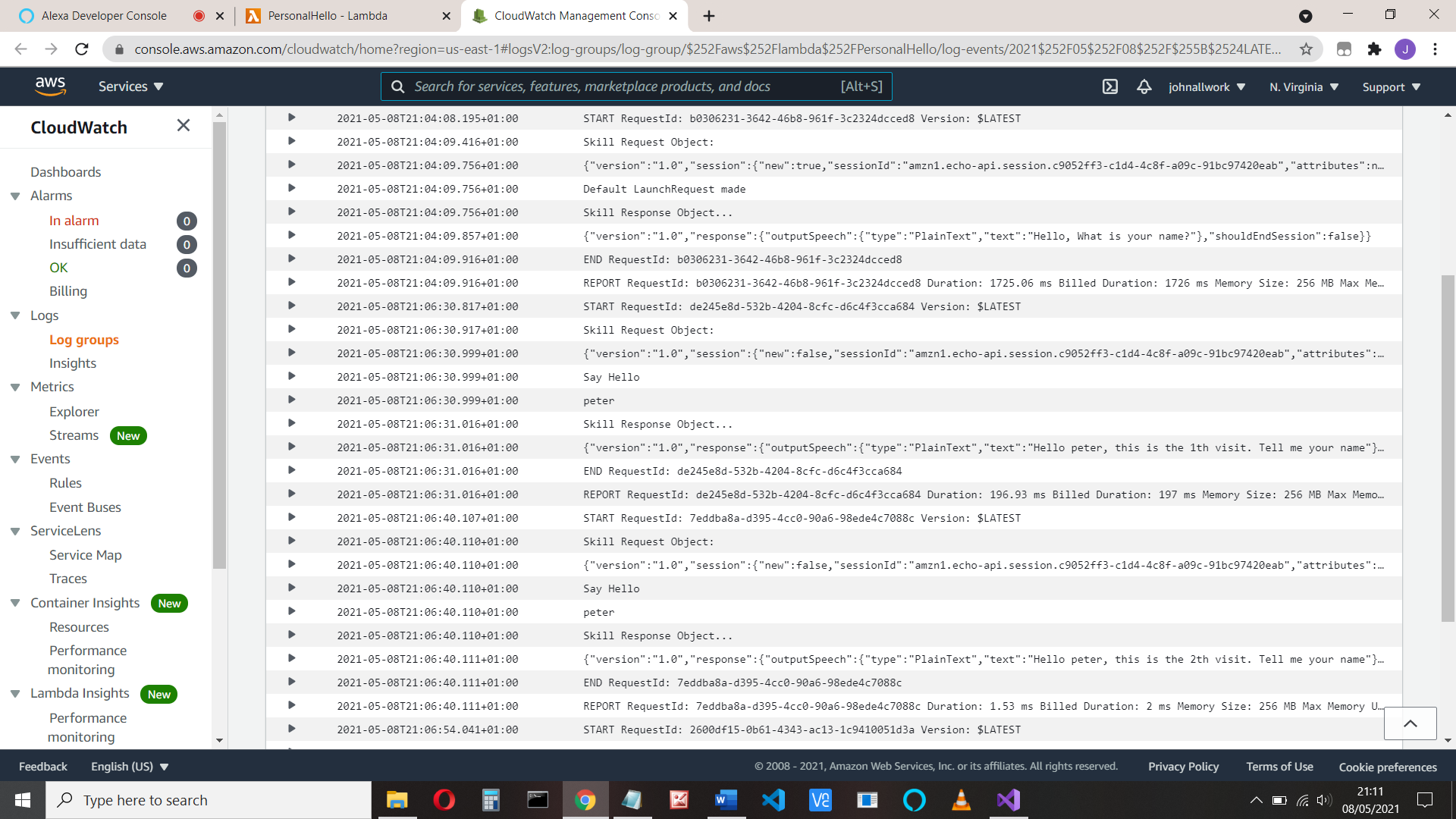
log.LogLine($"Skill Request Object:");

log.LogLine(JsonConvert.SerializeObject(input));

and

log.LogLine($"Skill Response Object...");

log.LogLine(JsonConvert.SerializeObject(response));



That’s it

Next time we’ll see how we pass data between our program and our device. We’ll play a guessing game.

Full code: - **Change the namespace If you haven’t names it ‘PersonalHello’**

using Amazon.Lambda.Core;

using Alexa.NET.Response;

using Alexa.NET.Request;

using Alexa.NET.Request.Type;

using Newtonsoft.Json;

using System.Collections.Generic;

using System;

// Assembly attribute to enable the Lambda function's JSON input

// to be converted into a .NET class.

[assembly: LambdaSerializer(typeof(Amazon.Lambda.Serialization.Json.JsonSerializer))]

namespace PersonalHello

{

public class Function

{

int count;

// string MYNAME = "myname";

string userName;

public SkillResponse FunctionHandler(SkillRequest input, ILambdaContext context)

{

SkillResponse response = new SkillResponse();

response.Response = new ResponseBody();

response.Response.ShouldEndSession = false;

IOutputSpeech innerResponse = null;

var log = context.Logger;

log.LogLine($"Skill Request Object:");

log.LogLine(JsonConvert.SerializeObject(input));

if (input.GetRequestType() == typeof(LaunchRequest))

{

count = 0;

userName = null;

log.LogLine($"Default LaunchRequest made");

innerResponse = new PlainTextOutputSpeech();

(innerResponse as PlainTextOutputSpeech).Text = "Hello, What is your name?";

}

else if (input.GetRequestType() == typeof(IntentRequest))

{

var intentRequest = (IntentRequest)input.Request;

switch (intentRequest.Intent.Name)

{

case "AMAZON.CancelIntent":

log.LogLine($"AMAZON.CancelIntent: send StopMessage");

innerResponse = new PlainTextOutputSpeech();

(innerResponse as PlainTextOutputSpeech).Text = "Cancelled";

response.Response.ShouldEndSession = true;

break;

case "AMAZON.StopIntent":

log.LogLine($"AMAZON.StopIntent: send StopMessage");

innerResponse = new PlainTextOutputSpeech();

(innerResponse as PlainTextOutputSpeech).Text = "Stopping";

response.Response.ShouldEndSession = true;

break;

case "AMAZON.HelpIntent":

log.LogLine($"AMAZON.HelpIntent: send HelpMessage");

innerResponse = new PlainTextOutputSpeech();

(innerResponse as PlainTextOutputSpeech).Text = "Help me!";

break;

case "HelloIntent":

count++;

log.LogLine($"Say Hello");

// they might want to change their name

if (intentRequest.Intent.Slots["myname"].Value == null)

// no name passed. Use anon if new, or use the name we know

{

if (userName == null) // we don't know their name, call them anon

{

userName = "anon";

}

}

else // myname has a value, use that

{

userName = intentRequest.Intent.Slots["myname"].Value;

}

log.LogLine(userName);

innerResponse = new PlainTextOutputSpeech();

(innerResponse as PlainTextOutputSpeech).Text = $"Hello {userName}, this is the {count}th visit. Tell me your name";

break;

default:

log.LogLine($"Unknown intent: " + intentRequest.Intent.Name);

innerResponse = new PlainTextOutputSpeech();

(innerResponse as PlainTextOutputSpeech).Text = "Unknown Intent";

break;

}

}

response.Response.OutputSpeech = innerResponse;

response.Version = "1.0";

log.LogLine($"Skill Response Object...");

log.LogLine(JsonConvert.SerializeObject(response));

return response;

}

}

}