

The background features a complex, abstract design composed of several concentric circles and arrows. The circles are rendered in white and light gray, creating a sense of depth and motion against a dark teal gradient background. Some circles have arrows pointing clockwise, while others have arrows pointing counter-clockwise, suggesting a dynamic or cyclical process.

# INTRODUCTION TO AWS ALEXA SKILL KIT

PRACTICAL EXAMPLE

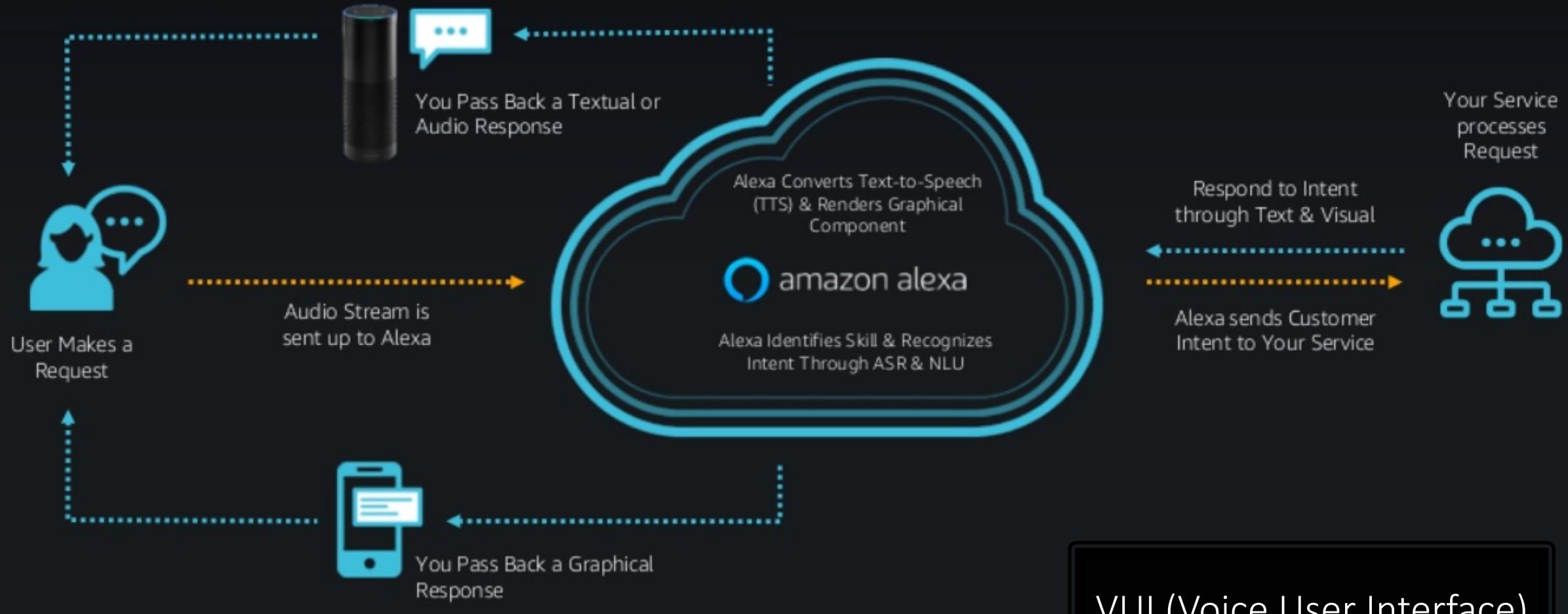
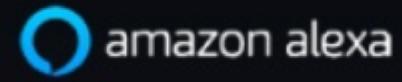
# WHO AM I?



- A developer in Ammeon,  
[www.ammeon.com](http://www.ammeon.com)
- 20 years of experience handling many position as IT Project Manager, System Administrator and Developer
- [linkedin.com/in/fcojperez](https://linkedin.com/in/fcojperez)
- Mail: [fcojperez@gmail.com](mailto:fcojperez@gmail.com)

# AGENDA

- Introduction
- Invocation, utterances, intents, Slots and AWS
- Types of skills
- Custom Skills archetype
  - A example of Skill archetype
- How to debug and testing, echosim.io
- ASK (Alexa Skill Kit)
  - Hands on
  - A example for ML Alexa Skill. ASK decision tree



# INVOCATION, INTENTS, UTTERANCES, SLOTS AND ENDPOINT/INTERFACE

# INVOCATION AND UTTERANCES

- Invocation
- Utterances



Invocation

Intents (5) + Add

- CouchPotatoIntent
- RecommendationIntent
  - salaryImportance
  - personality
  - preferredSpecies

For example, if the invocation name is "daily horoscopes", users can say:

User: Alexa, ask daily horoscopes for the horoscope for Gemini

Skill Invocation Name ⓘ

decision tree

Tip: Add my for yours unpublished skills

Invocation: A way for identifying skills

Utterance: The expression used for a invocation or intention

# INTENTS

Default  
Intents

Developed  
Intents

The screenshot shows the AWS Lambda console interface. On the left, there's a sidebar with links to Dashboard, Code Editor, and Intents (5). Below these are five listed intents: AMAZON.CancelIntent (required), AMAZON.HelpIntent (required), AMAZON.StopIntent (required), CouchPotatoIntent, and RecommendationIntent. An orange brace on the left groups the last three intents under the heading 'Developed Intents'. To the right, the 'CouchPotatoIntent' configuration page is displayed. It includes sections for Sample Utterances (with entries like "lazy" and "couch potato") and Intent confirmation (optional). A large orange callout box at the bottom defines an intent as "What the speaker want to ask or do or get".

CouchPotatoIntent

Sample Utterances (2) ?

What might a user say to invoke this intent? +

"lazy"

"couch potato"

Intent confirmation (optional) ?

Intent: What the speaker want to ask or do or get

# SLOTS

## Slot Types

[+ Add Slot Type](#)

Filter Slot Types

| NAME               | SLOT VALUES | TYPE   | ACTIONS                                       |
|--------------------|-------------|--------|---|
| articleType        | 2           | Custom | <a href="#">Edit</a>   <a href="#">Delete</a> |
| bloodToleranceType | 2           | Custom | <a href="#">Edit</a>   <a href="#">Delete</a> |
| IAmType            | 1           | Custom | <a href="#">Edit</a>   <a href="#">Delete</a> |
| iSubjectType       | 2           | Custom | <a href="#">Edit</a>   <a href="#">Delete</a> |
| personalityType    | 2           | Custom | <a href="#">Edit</a>   <a href="#">Delete</a> |

Slots: information passed as parameter by an utterance to an intent

# ENDPOINT/INTERFACES

## Interfaces



Enabling interfaces may add additional required intents to your interaction model. You will need to BOTH save interface changes and re-build your model for any updates to take effect.

| NAME              | DESCRIPTION  |                          |
|-------------------|--|--------------------------|
| Audio Player      | The AudioPlayer interface provides directives and requests for streaming audio and monitoring playback progression. <a href="#">Learn more</a> about the Audio Player Interface. | <input type="checkbox"/> |
| Display Interface | Echo Show allows skill developers to create skills for Alexa that use both screen and voice interaction. <a href="#">Learn more</a> about the Display Interface.                 | <input type="checkbox"/> |
| Video App         | The VideoApp interface provides the VideoApp.Launch directive for streaming native video files in Echo Show. <a href="#">Learn more</a> about the VideoApp Interface.            | <input type="checkbox"/> |

Interface: the content for some skills  
Endpoint: the logic for some intents

## Endpoint



The Endpoint will receive POST requests when a user interacts with your Alexa Skill. The request body contains parameters that your service can use to perform logic and generate a JSON-formatted response. Learn more about Lambda endpoints [here](#). You can host your own HTTPS web service endpoint as long as the service meets the requirements described [here](#).

### Service Endpoint Type

Select how you will host your skill's service endpoint.

AWS Lambda ARN (Recommended)

Your Skill ID ?

amzn1.ask.skill.1b4d05d2-d2a7-49db-b4a3-bbd988326f1d

[Copy to Clipboard](#)

Default Region  
?  
(Required)

arn:aws:lambda:us-east-1:061737249886:function:ask-cu

North America  
?  
(Optional)

arn:aws:lambda:us-east-1:<aws\_account\_id>:function:<lai>

Europe and India  
?  
(Optional)

arn:aws:lambda:eu-west-1:<aws\_account\_id>:function:<la>

Far East  
?  
(Optional)

arn:aws:lambda:ap-northeast-1:<aws\_account\_id>:func

HTTPS ?

# ENDPOINT, LAMBDA

Lambda > Functions > myStockTrackerDialog

ARN - arn:aws:lambda:eu-west-1:061737249886:function:myStockTrackerDialog

myStockTrackerDialog

Throttle Qualifiers Actions Test StartSession Test Save

Configuration Monitoring

Designer

Add triggers Click on a trigger from the list below to add it to your function.

API Gateway AWS IoT Alexa Skills Kit CloudWatch Events CloudWatch Logs

Alexa Skills Kit

Amazon CloudWatch Logs

Amazon DynamoDB

Resources the function's role has access to will be shown here

Add triggers from the list on the left

Function code

Code entry type Edit code inline Runtime Node.js 6.10 Handler index.handler

File Edit Find View Goto Tools Window

index.js logging.js speech.js callbackMock.js sessionMock.js

Lambda ARN

Endpoint

The Endpoint will receive POST requests when a user interacts with your Alexa Skill. The request body contains parameters that your service can use to perform logic and generate a JSON-formatted response. Learn more about Lambda endpoints [here](#). You can host your own HTTPS web service endpoint as long as the service meets the requirements described [here](#).

Service Endpoint Type

Select how you will host your skill's service endpoint.

AWS Lambda ARN (Recommended)

Your Skill ID [Copy to Clipboard](#)

Default Region (Required)

arn:aws:lambda:us-east-1:061737249886:function:ask-cu

# TYPES OF SKILLS

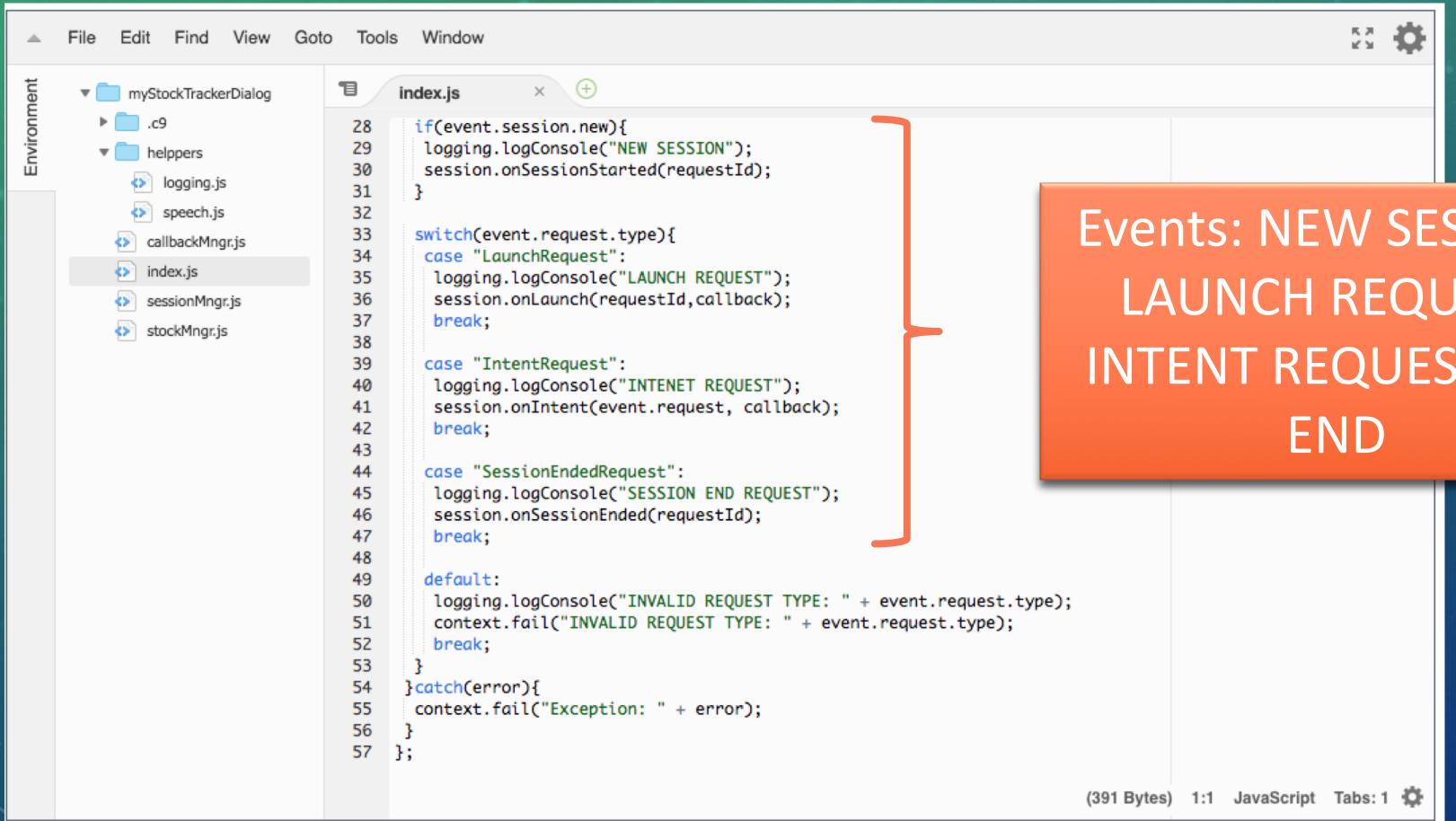
# TYPES OF SKILLS

|                    | Custom    | Smart Home         | Flash Briefing | Video          |
|--------------------|-----------|--------------------|----------------|----------------|
| Utterance          | Developer | Amazon Alexa       | Amazon Alexa   | Amazon Alexa   |
| Intent             | Developer | Developer          | Amazon Alexa   | Developer      |
| Properties (Slots) | Anything  | Appliance Specific | Content Needed | Device/Service |
| Uses Case          | Anything  | Smart Home Devices | Short Content  | Video Content  |

# CUSTOM SKILL ARCHETYPE



# CUSTOM SKILL ARCHETYPE



The screenshot shows a code editor interface with a toolbar at the top and a sidebar labeled "Environment" on the left. The sidebar contains a file tree with the following structure:

- myStockTrackerDialog
- .c9
- helpers
  - logging.js
  - speech.js
- callbackMngr.js
- index.js (selected)
- sessionMngr.js
- stockMngr.js

The main editor window displays the contents of the selected file, `index.js`:

```
28 if(event.session.new){  
29   logging.logConsole("NEW SESSION");  
30   session.onSessionStarted(requestId);  
31 }  
32  
33 switch(event.request.type){  
34   case "LaunchRequest":  
35     logging.logConsole("LAUNCH REQUEST");  
36     session.onLaunch(requestId,callback);  
37     break;  
38  
39   case "IntentRequest":  
40     logging.logConsole("INTENT REQUEST");  
41     session.onIntent(event.request, callback);  
42     break;  
43  
44   case "SessionEndedRequest":  
45     logging.logConsole("SESSION END REQUEST");  
46     session.onSessionEnded(requestId);  
47     break;  
48  
49   default:  
50     logging.logConsole("INVALID REQUEST TYPE: " + event.request.type);  
51     context.fail("INVALID REQUEST TYPE: " + event.request.type);  
52     break;  
53 }  
54 }catch(error){  
55   context.fail("Exception: " + error);  
56 }  
57 };
```

At the bottom of the editor, status indicators show "(391 Bytes) 1:1 JavaScript Tabs: 1". A red curly brace is positioned to the right of the code editor, pointing towards the orange callout box.

Events: NEW SESSION,  
LAUNCH REQUEST,  
INTENT REQUEST and  
END

# CUSTOM SKILL ARCHETYPE

| Module             | Function   |
|--------------------|--|
| Helpers/logging.js | Console logging and future logging information                           |
| Helpers/speech.js  | Manage the speech building process                                       |
| callbackMngr.js    | Manage the Alexa Callback process  |
| Index.js           | Depends on the event, context and callback address the modulo and method |
| sessionMngr.js     | Manage the event session   |
| stockMngr.js       | Modulo for managing the Web Service request                              |



<https://github.com/fcojperez/myStockTrackerDialog>

# HOW TO DEBUG AND TESTING, ECHOSIM.IO

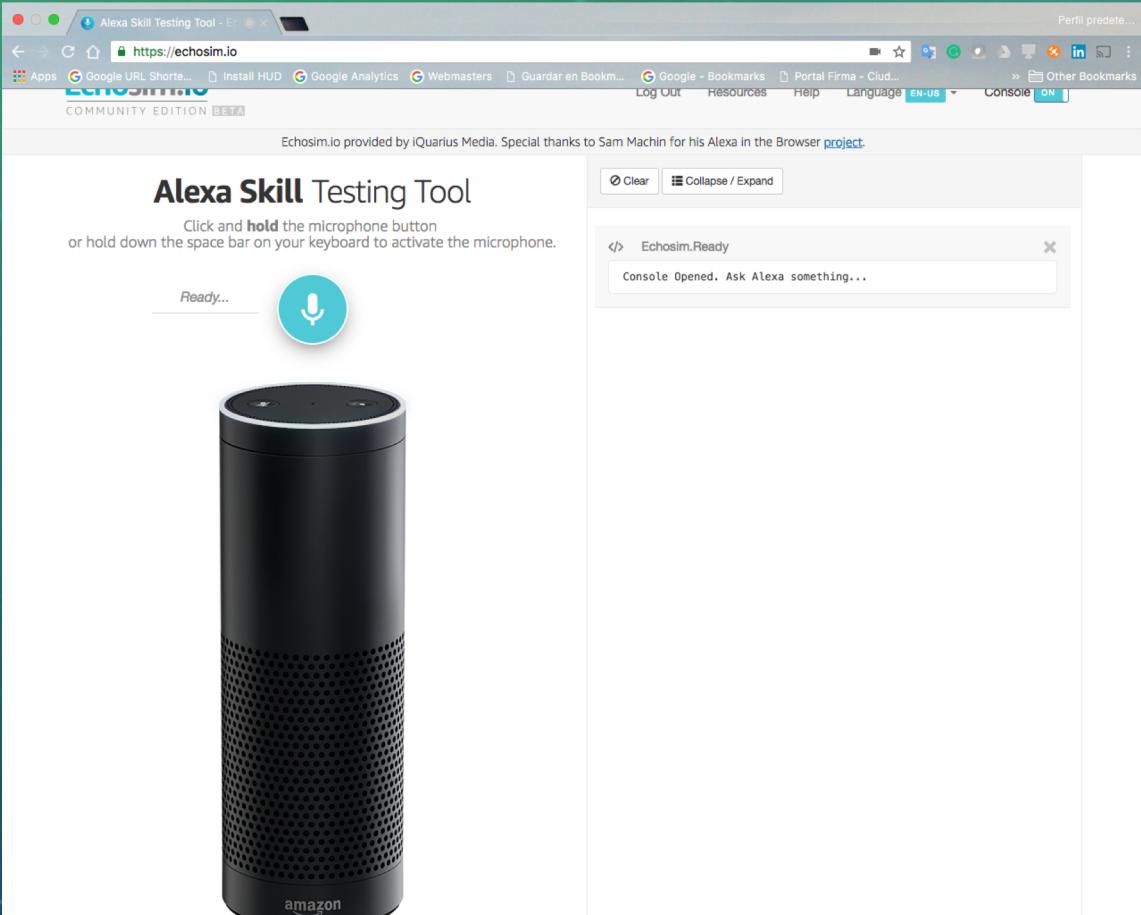
# DEBUGGING: AMAZON CLOUDWATCH LOGS

| Filter events  |   | all 2018-03-23 (00:00:00) - 2018-03-24 (00:00:00) |
|--|---|---|
| Time (UTC +00:00)  | Message   | Show in stream                                    |
| 2018-03-23   |   |   |
| ▶ 20:19:13   | 2018-03-23T20:19:13.148Z 73dc3e78-2ed7-11e8-9301-5ba568c5ed7d All slots filled: StockMng            | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.148Z 73dc3e78-2ed7-11e8-9301-5ba568c5ed7d In function stockMng                  | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.148Z 73dc3e78-2ed7-11e8-9301-5ba568c5ed7d All slots filled: StockMng            | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.148Z 73dc3e78-2ed7-11e8-9301-5ba568c5ed7d Values mapped for s                   | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.148Z 73dc3e78-2ed7-11e8-9301-5ba568c5ed7d Path string: /historic                | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.783Z 73dc3e78-2ed7-11e8-9301-5ba568c5ed7d Received json response                | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.824Z 73dc3e78-2ed7-11e8-9301-5ba568c5ed7d Processing response                   | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.825Z 73dc3e78-2ed7-11e8-9301-5ba568c5ed7d amazon high price w                   | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | END RequestId: 73dc3e78-2ed7-11e8-9301-5ba568c5ed7d   | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | REPORT RequestId: 73dc3e78-2ed7-11e8-9301-5ba568c5ed7d Duration: 678.45 ms Billed                   | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | START RequestId: 745a6f48-2ed7-11e8-9961-993ece3618c5 Version: \$LATEST                             | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.976Z 745a6f48-2ed7-11e8-9961-993ece3618c5 INTERNET REQUEST                      | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.976Z 745a6f48-2ed7-11e8-9961-993ece3618c5 In function sessionMng                | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.976Z 745a6f48-2ed7-11e8-9961-993ece3618c5 GetStockInfo intent : [object Object] | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.976Z 745a6f48-2ed7-11e8-9961-993ece3618c5 In function stockMng                  | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.976Z 745a6f48-2ed7-11e8-9961-993ece3618c5 In function stockMng                  | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.976Z 745a6f48-2ed7-11e8-9961-993ece3618c5 Dialog state: COMPLETED               | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.976Z 745a6f48-2ed7-11e8-9961-993ece3618c5 Intent: [object Object]               | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| 2018-03-23T20:19:13.976Z 745a6f48-2ed7-11e8-9961-993ece3618c5 Intent: [object Object]  |   |   |
| ▶ 20:19:13   | 2018-03-23T20:19:13.976Z 745a6f48-2ed7-11e8-9961-993ece3618c5 All slots filled: StockMng            | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| 2018-03-23T20:19:13.976Z 745a6f48-2ed7-11e8-9961-993ece3618c5 All slots filled: StockName: amazon, StockDate: 2018-03-12 and PriceType: high |   |   |
| ▶ 20:19:13   | 2018-03-23T20:19:13.976Z 745a6f48-2ed7-11e8-9961-993ece3618c5 In function stockMng                  | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.976Z 745a6f48-2ed7-11e8-9961-993ece3618c5 All slots filled: StockMng            | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.976Z 745a6f48-2ed7-11e8-9961-993ece3618c5 Values mapped for s                   | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:13   | 2018-03-23T20:19:13.976Z 745a6f48-2ed7-11e8-9961-993ece3618c5 Path string: /historic                | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:14   | 2018-03-23T20:19:14.312Z 745a6f48-2ed7-11e8-9961-993ece3618c5 Received json response                | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:14   | 2018-03-23T20:19:14.313Z 745a6f48-2ed7-11e8-9961-993ece3618c5 Processing response                   | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:14   | 2018-03-23T20:19:14.313Z 745a6f48-2ed7-11e8-9961-993ece3618c5 amazon high price w                   | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:14   | END RequestId: 745a6f48-2ed7-11e8-9961-993ece3618c5   | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |
| ▶ 20:19:14   | REPORT RequestId: 745a6f48-2ed7-11e8-9961-993ece3618c5 Duration: 607.07 ms Billed                   | <a href="#">2018/03/23[\$LATEST]0cda0f4...</a>    |

As best practice logging:

- At the beginning of each module / function
- Slots Values
- Slots Values processed
- Exceptions

# ECHOSIM.IO



As best practice test:

- Invocation name
- Utterance

# ASK (ALEXA SKILL KIT)

# ASK (ALEXA SKILL KIT)

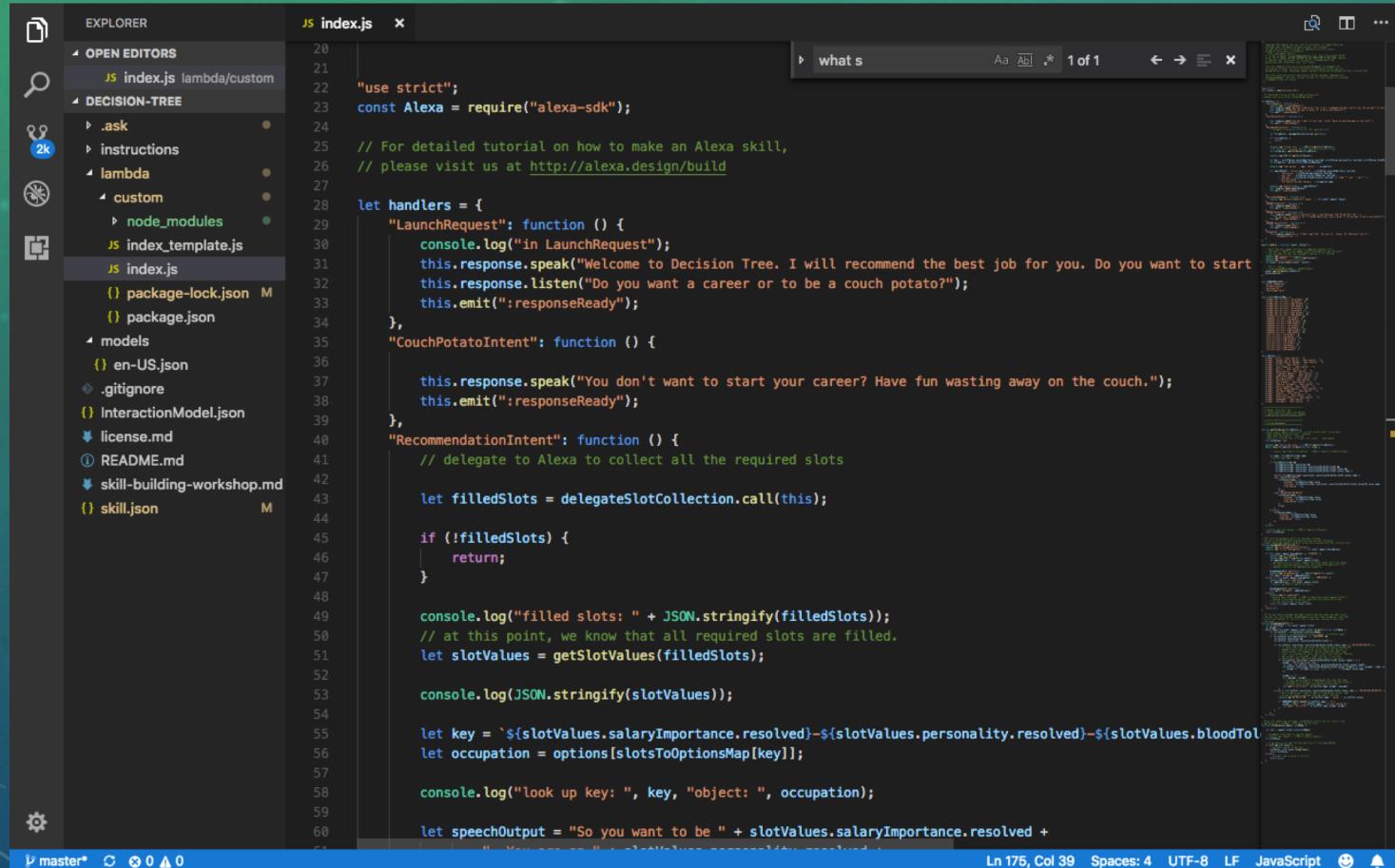
```
Last login: Fri Mar 23 23:29:57 on ttys002
[Host001:~ fcojperez$ ask new --template
? List of templates you can chose (Use arrow keys)
➤ City Guide
  Decision Tree
  Fact
  Feed
  High Low Game
  How To
  Pet Match
(Move up and down to reveal more choices)
```

## ask new –template

### Template List:

- City Guide
- Decision Tree
- Fact
- Feed
- High Low Game
- How to
- Pet Match
- ....

# DECISION TREE



index.js

```
20
21
22 "use strict";
23 const Alexa = require("alexa-sdk");
24
25 // For detailed tutorial on how to make an Alexa skill,
26 // please visit us at http://alexa.design/build
27
28 let handlers = {
29   "LaunchRequest": function () {
30     console.log("in LaunchRequest");
31     this.response.speak("Welcome to Decision Tree. I will recommend the best job for you. Do you want to start");
32     this.response.listen("Do you want a career or to be a couch potato?");
33     this.emit(":responseReady");
34   },
35   "CouchPotatoIntent": function () {
36     this.response.speak("You don't want to start your career? Have fun wasting away on the couch.");
37     this.emit(":responseReady");
38   },
39   "RecommendationIntent": function () {
40     // delegate to Alexa to collect all the required slots
41
42     let filledSlots = delegateSlotCollection.call(this);
43
44     if (!filledSlots) {
45       return;
46     }
47
48     console.log("filled slots: " + JSON.stringify(filledSlots));
49     // at this point, we know that all required slots are filled.
50     let slotValues = getSlotValues(filledSlots);
51
52     console.log(JSON.stringify(slotValues));
53
54     let key = `${slotValues.salaryImportance.resolved}-${slotValues.personality.resolved}-${slotValues.bloodType.resolved}`;
55     let occupation = options[slotsToOptionsMap[key]];
56
57     console.log("look up key: ", key, "object: ", occupation);
58
59     let speechOutput = "So you want to be " + slotValues.salaryImportance.resolved +
60     " " + slotValues.personality.resolved + " " + slotValues.bloodType.resolved + " " + occupation;
61
62     this.response.speak(speechOutput);
63     this.response.emit(":responseReady");
64   }
65 }
```

index.js

index\_template.js

index.json

package-lock.json

package.json

models

en-US.json

.gitignore

InteractionModel.json

license.md

README.md

skill-building-workshop.md

skill.json

master\* 0 0 ▲ 0

Ln 175, Col 39 Spaces: 4 UTF-8 LF JavaScript

## Slot Types / salaryImportanceType

### Slot Values (3)

| Value       | ID (Optional) | Synonyms (Optional) |
|-------------|---------------|---------------------|
| unimportant | Enter ID      | Add synonym         |
| somewhat    | Enter ID      | Add synonym         |
| very        | Enter ID      | Add synonym         |

Search

Enter a new value for this slot type

+ money is evil money is not important not a sell out nope don't care about money

+ just enough somewhat important not top priority support my family

+ high salary filthy rich billionaire important nothing more important