

1. Guitar Tuner

■ Logging ■ Persisted State ■ Helper functions ■ Alexa boilerplate

```
1 'use strict';
2 const Alexa = require('ask-sdk-v1adapter');
3
4 const handlers = {
5   'LaunchRequest': function () {
6     logEvent(this.event);
7
8     this.emit(':ask', 'Welcome to Guitar Tuner, what note would you like me to play?');
9   },
10  'PlayNoteIntent': function () {
11    logEvent(this.event);
12
13    let note = getSlotValue(this.event, 'note');
14    setAttrValue(this.event, 'note', note);
15
16    this.emit(':ask', 'What pitch would you like for ' + note + '?');
17  },
18  'SpecifyPitchIntent': function () {
19    logEvent(this.event);
20
21    let note = getAttrValue(this.event, 'note');
22    let pitch = getSlotValue(this.event, 'pitch');
23
24    this.emit(':ask', getNotePitchOutput(note, pitch));
25  },
26  'PlayNoteWithPitchIntent': function () {
27    logEvent(this.event);
28
29    let note = getSlotValue(this.event, 'note');
30    let pitch = getSlotValue(this.event, 'pitch');
31
32    this.emit(':ask', getNotePitchOutput(note, pitch));
33  },
34  'AMAZON.HelpIntent': function () {
35    // This is triggered when users say "Help"
36  },
37  'AMAZON.CancelIntent': function () {
38    // This is triggered when users say "Cancel"
39  },
40  'AMAZON.StopIntent': function () {
41    // This is triggered when users say "Stop"
42  },
43  'AMAZON.NavigateHomeIntent': function () {
44    // This is triggered when users say "Navigate Home"
45  },
46  'AMAZON.FallbackIntent': function () {
47    // This is triggered when users say something that doesn't map to an intent
```

```

48     this.emit(':tell', 'Sorry, I couldn\'t understand you');
49 },
50 'SessionEndedRequest': function () {
51     // This is triggered when users say "Exit"
52 }
53 };
54
55 function getNotePitchOutput(note, pitch) {
56     let output = 'You requested a note ' + note + ' with pitch ' + pitch;
57     console.log('Output: ' + output);
58     return output;
59 }
60
61 function getAttrValue(event, attrName) {
62     var attrValue = event.session.attributes[attrName];
63     console.log('Session has attribute ' + attrName + ': ' + attrValue);
64     return attrValue;
65 }
66
67 function setAttrValue(event, attrName, attrValue) {
68     event.session.attributes[attrName] = attrValue;
69     console.log('Saved session attribute ' + attrName + ': ' + attrValue);
70 }
71
72 function getSlotValue(event, slotName) {
73     let slot = event.request.intent.slots[slotName];
74     let slotValue;
75     try {
76         // Resolve to the canonical value if available
77         slotValue = slot.resolutions.resolutionsPerAuthority[0].values[0].value.name;
78     } catch(err) {
79         // Otherwise fallback to the raw text
80         console.log(err.message);
81         slotValue = slot.value;
82     }
83
84     console.log('User provided ' + slotName + ': ' + slotValue);
85     return slotValue;
86 }
87
88 function logEvent(event) {
89     console.log('Request:');
90     console.log(JSON.stringify(event.request));
91     console.log('Session:');
92     console.log(JSON.stringify(event.session));
93 }
94
95 exports.handler = function(event, context, callback) {
96     const alexa = Alexa.handler(event, context, callback);
97     alexa.registerHandlers(handlers);
98     alexa.execute();
99 };

```

2. High Low Game

Logging Input validation Persisted state Helper functions Alexa boilerplate

```
1 'use strict';
2 const Alexa = require('ask-sdk-v1adapter');
3
4 const ALLOWED_HINTS = [ 'low', 'high' ];
5
6 function avg(n1, n2) {
7     return Math.floor((n1 + n2) / 2);
8 }
9
10 const handlers = {
11     'LaunchRequest': function () {
12         logEvent(this.event);
13
14         this.emit(':ask', 'Welcome to High Low Game. Think of a number between 1 and 100,
15         and say "start" when ready. ');
16     },
17     'StartIntent': function () {
18         logEvent(this.event);
19
20         let minGuess = 1;
21         let maxGuess = 100;
22         let guess = avg(minGuess, maxGuess);
23
24         setAttrValue(this.event, 'guessesLeft', 4);
25         setAttrValue(this.event, 'minGuess', minGuess);
26         setAttrValue(this.event, 'maxGuess', maxGuess);
27         setAttrValue(this.event, 'lastGuess', guess);
28
29         this.emit(':ask', produceGuessOutput(guess));
30     },
31     'HintIntent': function () {
32         logEvent(this.event);
33
34         let lastGuess = getAttrValue(this.event, 'lastGuess');
35         let hint = getSlotValue(this.event, 'hint');
36         if (!ALLOWED_HINTS.includes(hint)) {
37             this.emit(':ask', 'Sorry, I didn\'t get your hint. Is ' + lastGuess + ' high,
38             low or correct? ');
39             return;
40         }
41
42         let guessesLeft = getAttrValue(this.event, 'guessesLeft');
43         if (guessesLeft === 0) {
44             this.emit(':tell', 'I lost. Good game, thank you for playing!');
45             return;
46         }
47         setAttrValue(this.event, 'guessesLeft', guessesLeft - 1);
```

```

46
47     let minGuess, maxGuess;
48
49     if (hint === 'low') {
50         minGuess = lastGuess;
51         setAttrValue(this.event, 'minGuess', minGuess);
52         maxGuess = getAttrValue(this.event, 'maxGuess');
53     } else {
54         minGuess = getAttrValue(this.event, 'minGuess');
55         maxGuess = lastGuess;
56         setAttrValue(this.event, 'maxGuess', maxGuess);
57     }
58
59     let guess = avg(minGuess, maxGuess);
60     setAttrValue(this.event, 'lastGuess', guess);
61     this.emit(':ask', produceGuessOutput(guess));
62 },
63 'CorrectIntent': function () {
64     logEvent(this.event);
65
66     this.emit(':tell', 'Hooray! Thank you for playing!');
67 },
68 'AMAZON.HelpIntent': function () {
69     // This is triggered when users say "Help"
70 },
71 'AMAZON.CancelIntent': function () {
72     // This is triggered when users say "Cancel"
73 },
74 'AMAZON.StopIntent': function () {
75     // This is triggered when users say "Stop"
76 },
77 'AMAZON.NavigateHomeIntent': function () {
78     // This is triggered when users say "Navigate Home"
79 },
80 'AMAZON.FallbackIntent': function () {
81     // This is triggered when users say something that doesn't map to an intent
82 },
83 'SessionEndedRequest': function () {
84     // This is triggered when users say "Exit"
85 }
86 };
87
88 function produceGuessOutput(guess) {
89     return guess + '. Is it high, low or correct?';
90 }
91
92 function getAttrValue(event, attrName) {
93     var attrValue = event.session.attributes[attrName];
94     console.log('Session has attribute ' + attrName + ': ' + attrValue);
95     return attrValue;
96 }
97








```

```

98 function setAttrValue(event, attrName, attrValue) {
99     event.session.attributes[attrName] = attrValue;
100     console.log('Saved session attribute ' + attrName + ': ' + attrValue);
101 }
102
103 function getSlotValue(event, slotName) {
104     let slot = event.request.intent.slots[slotName];
105     let slotValue;
106     try {
107         // Resolve to the canonical value if available
108         slotValue = slot.resolutions.resolutionsPerAuthority[0].values[0].value.name;
109     } catch(err) {
110         // Otherwise fallback to the raw text
111         console.log(err.message);
112         slotValue = slot.value;
113     }
114
115     console.log('User provided ' + slotName + ': ' + slotValue);
116     return slotValue;
117 }
118
119 function logEvent(event) {
120     console.log('Request:');
121     console.log(JSON.stringify(event.request));
122     console.log('Session:');
123     console.log(JSON.stringify(event.session));
124 }
125
126 exports.handler = function(event, context, callback) {
127     const alexa = Alexa.handler(event, context, callback);
128     alexa.registerHandlers(handlers);
129     alexa.execute();
130 };

```

3. Artistic Joke

 Logging	 Input validation	 Random choice	 Dictionary lookup
 Modified speech (SSML)	 Helper functions	 Alexa boilerplate	

```
1 'use strict';
2 const Alexa = require('ask-sdk-v1adapter');
3
4 const MODIFIER_TO_PROSODY = {
5   'quiet': 'volume="x-soft"',
6   'loud': 'volume="x-loud"',
7   'quick': 'rate="x-fast"',
8   'slow': 'rate="x-slow"',
9   'high': 'pitch="x-high"',
10  'low': 'pitch="x-low"'
11 };
12
13 const ALLOWED_MODIFIERS = Object.keys(MODIFIER_TO_PROSODY);
14
15 const JOKES = [
16   'Today at the bank, an old lady asked me to help check her balance. So I pushed her over.',
17   'I couldn\'t figure out why the baseball kept getting larger. Then it hit me.',
18   'I told my girlfriend she drew her eyebrows too high. She seemed surprised.',
19   'My dog used to chase people on a bike a lot. It got so bad, finally I had to take his bike away.',
20   'I\'m so good at sleeping. I can do it with my eyes closed.',
21   'My boss told me to have a good day... so I went home.',
22   'A woman walks into a library and asked if they had any books about paranoia. The librarian says "They\'re right behind you!"',
23   'The other day, my wife asked me to pass her lipstick but I accidentally passed her a glue stick. She still isn\'t talking to me.',
24   'Why do blind people hate skydiving? It scares the hell out of their dogs.',
25   'When you look really closely, all mirrors look like eyeballs.'
26 ];
27
28 function getRandomInt(max) {
29   return Math.floor(Math.random() * Math.floor(max));
30 }
31
32 function pickAJoke() {
33   return JOKES[getRandomInt(JOKES.length)];
34 }
35
36 const handlers = {
37   'LaunchRequest': function () {
38     logEvent(this.event);
39   }
```

```

40     this.emit(':ask', 'What kind of joke do you want to hear?');
41 },
42 'TellAJokeIntent': function () {
43     logEvent(this.event);
44
45     let joke = pickAJoke();
46     this.emit(':ask', joke);
47 },
48 'WhisperAJokeIntent': function () {
49     logEvent(this.event);
50
51     let joke = pickAJoke();
52
53     // Docs: https://developer.amazon.com/docs/custom-skills/speech-synthesis-
markup-language-ssml-reference.html'
54     let ssmlResponse = '<amazon:effect name="whispered">' + joke +
'</amazon:effect>';
55     this.emit(':ask', ssmlResponse);
56 },
57 'ModifiedJokeIntent': function () {
58     logEvent(this.event);
59
60     let modifier = getSlotValue(this.event, 'modifier');
61     if (!ALLOWED_MODIFIERS.includes(modifier)) {
62         this.emit(':ask', 'I don\'t know that kind of jokes');
63         return;
64     }
65
66     let joke = pickAJoke();
67     let prosody = MODIFIER_TO_PROSODY[modifier];
68
69     // Example: <prosody pitch="high">Pinochio drowned</prosody>
70     let ssmlResponse = '<prosody ' + prosody + '>' + joke + '</prosody>';
71     this.emit(':ask', ssmlResponse);
72 },
73 'AMAZON.HelpIntent': function () {
74     // This is triggered when users say "Help"
75 },
76 'AMAZON.CancelIntent': function () {
77     // This is triggered when users say "Cancel"
78 },
79 'AMAZON.StopIntent': function () {
80     // This is triggered when users say "Stop"
81 },
82 'AMAZON.NavigateHomeIntent': function () {
83     // This is triggered when users say "Navigate Home"
84 },
85 'AMAZON.FallbackIntent': function () {
86     // This is triggered when users say something that doesn't map to an intent
87 },
88 'SessionEndedRequest': function () {
89     // This is triggered when users say "Exit"

```

```

90     }
91 };
92
93 function getSlotValue(event, slotName) {
94     let slot = event.request.intent.slots[slotName];
95     let slotValue;
96     try {
97         // Resolve to the canonical value if available
98         slotValue = slot.resolutions.resolutionsPerAuthority[0].values[0].value.name;
99     } catch(err) {
100         // Otherwise fallback to the raw text
101         console.log(err.message);
102         slotValue = slot.value;
103     }
104
105     console.log('User provided ' + slotName + ': ' + slotValue);
106     return slotValue;
107 }
108
109 function getAttrValue(event, attrName) {
110     var attrValue = event.session.attributes[attrName];
111     console.log('Session has attribute ' + attrName + ': ' + attrValue);
112     return attrValue;
113 }
114
115 function setAttrValue(event, attrName, attrValue) {
116     event.session.attributes[attrName] = attrValue;
117     console.log('Saved session attribute ' + attrName + ': ' + attrValue);
118 }
119
120 function logEvent(event) {
121     console.log('Request:');
122     console.log(JSON.stringify(event.request));
123     console.log('Session:');
124     console.log(JSON.stringify(event.session));
125 }
126
127 exports.handler = function(event, context, callback) {
128     const alexa = Alexa.handler(event, context, callback);
129     alexa.registerHandlers(handlers);
130     alexa.execute();
131 };

```


4. Seattle Getaway

Logging

Web request

Helper functions

Alexa boilerplate

```
1 'use strict';
2 const Alexa = require('ask-sdk-v1adapter');
3 const https = require('https');
4
5 function distanceCallback(responseString, callbackParam) {
6     let city = callbackParam[0];
7     let emit = callbackParam[1];
8     let responseJson = JSON.parse(responseString);
9     let durationText = responseJson.rows[0].elements[0].duration.text;
10    emit(':ask', 'Drive to ' + city + ' is going to be ' + durationText);
11 }
12
13 const handlers = {
14     'LaunchRequest': function () {
15         logEvent(this.event);
16
17         this.emit(':ask', 'Which city would you like to go to?');
18     },
19     'PickCityIntent': function () {
20         logEvent(this.event);
21
22         let city = getSlotValue(this.event, 'city');
23         let encodedCity = encodeURIComponent(city);
24
25         httpGet(
26             'maps.googleapis.com',
27             '/maps/api/distancematrix/json?origins=Seattle,WA&destinations=' + encodedCity
28             + '&key=AIzaSyB4nTfp2M_t2L_zKj4Z4Wpb61mGs_wOQAw',
29             distanceCallback,
30             [city, this.emit]);
31
32         // httpGet() returns before the request is finished
33         // We don't know the response here, callback will know it
34         // Alexa will wait till callback calls emit()
35     },
36     'AMAZON.HelpIntent': function () {
37         // This is triggered when users say "Help"
38     },
39     'AMAZON.CancelIntent': function () {
40         // This is triggered when users say "Cancel"
41     },
42     'AMAZON.StopIntent': function () {
43         // This is triggered when users say "Stop"
44     },
45     'AMAZON.NavigateHomeIntent': function () {
46         // This is triggered when users say "Navigate Home"
```

```

47 'AMAZON.FallbackIntent': function () {
48     // This is triggered when users say something that doesn't map to an intent
49 },
50 'SessionEndedRequest': function () {
51     // This is triggered when users say "Exit"
52 }

```

```

53 };

```

```

54

```

```

55 function httpGet(host, path, callback, callbackParam) {
56     console.log('Outgoing request host: ' + host);
57     console.log('Outgoing request path: ' + path);
58     console.log('Outgoing request callback param: ' + callbackParam);
59
60     let options = {
61         host: host,
62         path: path,
63         method: 'GET',
64     };
65
66     let req = https.request(options, res => {
67         res.setEncoding('utf8');
68         let responseString = '';
69
70         //accept incoming data asynchronously
71         res.on('data', chunk => {
72             responseString = responseString + chunk;
73         });
74
75         //return the data when streaming is complete
76         res.on('end', () => {
77             console.log('Received response: ' + responseString);
78             callback(responseString, callbackParam);
79         });
80
81     });
82     req.end();
83 }

```

```

84

```

```

85 function getSlotValue(event, slotName) {
86     let slot = event.request.intent.slots[slotName];
87     let slotValue;
88     try {
89         // Resolve to the canonical value if available
90         slotValue = slot.resolutions.resolutionsPerAuthority[0].values[0].value.name;
91     } catch(err) {
92         // Otherwise fallback to the raw text
93         console.log(err.message);
94         slotValue = slot.value;
95     }
96
97     console.log('User provided ' + slotName + ': ' + slotValue);
98     return slotValue;

```

```

99 }
100
101 function getAttrValue(event, attrName) {
102     var attrValue = event.session.attributes[attrName];
103     console.log('Session has attribute ' + attrName + ': ' + attrValue);
104     return attrValue;
105 }
106
107 function setAttrValue(event, attrName, attrValue) {
108     event.session.attributes[attrName] = attrValue;
109     console.log('Saved session attribute ' + attrName + ': ' + attrValue);
110 }
111
112 function logEvent(event) {
113     console.log('Request:');
114     console.log(JSON.stringify(event.request));
115     console.log('Session:');
116     console.log(JSON.stringify(event.session));
117 }
118
119 exports.handler = function(event, context, callback) {
120     const alexa = Alexa.handler(event, context, callback);
121     alexa.registerHandlers(handlers);
122     alexa.execute();
123 };

```