const Alexa = require('ask-sdk-core');

const LaunchRequestHandler = {

canHandle(handlerInput) {

return Alexa.getRequestType(handlerInput.requestEnvelope) === 'LaunchRequest';

},

handle(handlerInput) {

const speakOutput = 'Welcome, you can say Hello or Help. Which would you like to try?';

return handlerInput.responseBuilder

.speak(speakOutput)

.reprompt(speakOutput)

.getResponse();

}

};

const WelcomeIntentHandler = {

canHandle(handlerInput) {

return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest' &&

Alexa.getIntentName(handlerInput.requestEnvelope) === 'WelcomeIntent';

},

handle(handlerInput) {

const greetingSlot = handlerInput.requestEnvelope.request.intent.slots.greeting.value;

let speakOutput = 'Hello, welcome to Alexa!';

if (greetingSlot) {

speakOutput = `Hi ${greetingSlot}, nice to welcome you!`;

}

return handlerInput.responseBuilder

.speak(speakOutput)

.reprompt('What else can I help you with?')

.getResponse();

}

};

// Function to sanitize or process the slot value

function sanitizeGreeting(greeting) {

// Example sanitization: trim whitespace and capitalize the first letter

return greeting.trim().charAt(0).toUpperCase() + greeting.slice(1).toLowerCase();

}

const HelloWorldIntentHandler = {

canHandle(handlerInput) {

return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'

&& Alexa.getIntentName(handlerInput.requestEnvelope) === 'HelloWorldIntent';

},

handle(handlerInput) {

const speakOutput = 'Hello World!';

return handlerInput.responseBuilder

.speak(speakOutput)

//.reprompt('add a reprompt if you want to keep the session open for the user to respond')

.getResponse();

}

};

const HelpIntentHandler = {

canHandle(handlerInput) {

return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'

&& Alexa.getIntentName(handlerInput.requestEnvelope) === 'AMAZON.HelpIntent';

},

handle(handlerInput) {

const speakOutput = 'You can say hello to me! How can I help?';

return handlerInput.responseBuilder

.speak(speakOutput)

.reprompt(speakOutput)

.getResponse();

}

};

const CancelAndStopIntentHandler = {

canHandle(handlerInput) {

return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'

&& (Alexa.getIntentName(handlerInput.requestEnvelope) === 'AMAZON.CancelIntent'

|| Alexa.getIntentName(handlerInput.requestEnvelope) === 'AMAZON.StopIntent');

},

handle(handlerInput) {

const speakOutput = 'Goodbye!';

return handlerInput.responseBuilder

.speak(speakOutput)

.getResponse();

}

};

const FallbackIntentHandler = {

canHandle(handlerInput) {

return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'

&& Alexa.getIntentName(handlerInput.requestEnvelope) === 'AMAZON.FallbackIntent';

},

handle(handlerInput) {

const speakOutput = 'Sorry, I don\'t know about that. Please try again.';

return handlerInput.responseBuilder

.speak(speakOutput)

.reprompt(speakOutput)

.getResponse();

}

};

const SessionEndedRequestHandler = {

canHandle(handlerInput) {

return Alexa.getRequestType(handlerInput.requestEnvelope) === 'SessionEndedRequest';

},

handle(handlerInput) {

console.log(`~~~~ Session ended: ${JSON.stringify(handlerInput.requestEnvelope)}`);

// Any cleanup logic goes here.

return handlerInput.responseBuilder.getResponse(); // notice we send an empty response

}

};

const IntentReflectorHandler = {

canHandle(handlerInput) {

return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest';

},

handle(handlerInput) {

const intentName = Alexa.getIntentName(handlerInput.requestEnvelope);

const speakOutput = `You just triggered ${intentName}`;

return handlerInput.responseBuilder

.speak(speakOutput)

//.reprompt('add a reprompt if you want to keep the session open for the user to respond')

.getResponse();

}

};

const ErrorHandler = {

canHandle() {

return true;

},

handle(handlerInput, error) {

const speakOutput = 'Sorry, I had trouble doing what you asked. Please try again.';

console.log(`~~~~ Error handled: ${JSON.stringify(error)}`);

return handlerInput.responseBuilder

.speak(speakOutput)

.reprompt(speakOutput)

.getResponse();

}

};

exports.handler = Alexa.SkillBuilders.custom()

.addRequestHandlers(

LaunchRequestHandler,

WelcomeIntentHandler, // Ensure this is included

HelloWorldIntentHandler,

HelpIntentHandler,

CancelAndStopIntentHandler,

FallbackIntentHandler,

SessionEndedRequestHandler,

IntentReflectorHandler) // Make sure it's last

.addErrorHandlers(

ErrorHandler)

.withCustomUserAgent('sample/hello-world/v1.2')

.lambda();