



Alexaハンズオントレーニング 初級編

ASK Developer Marketing, Amazon Japan

Version 2.8, 2018/10/25

目次

はじめに	1
トレーニングの学習目標	2
課題実習の環境要件	3
ASK SDK for Node.js のバージョン	3
ダウンロード	4
最新版教材のダウンロード	4
サンプルコードのダウンロード	4
プログラミング経験について	5
1. 初めて作るカスタムスキル	6
1.1. この実習の学習目標	6
1.2. 対話モデルのデザイン	7
1.3. エンドポイントの開発	19
1.4. エンドポイントを登録する	28
1.5. テストする	30
2. インテントの追加	38
2.1. この実習の学習目標	38
2.2. 対話モデルの編集	38
2.3. テストする	44
3. スロットの使用	46
3.1. この実習の学習目標	46
3.2. 対話モデルの編集	46
3.3. Lambdaのコードを改良する	52
3.4. テストする	53
4. 発話のバリエーション	55
4.1. この実習の学習目標	55
4.2. 対話モデルの編集	55
4.3. Lambdaのコードを改良する	57
4.4. テストする	59
5. セッションアトリビュート	62
5.1. この実習の学習目標	62
5.2. 会話のサンプル	62
5.3. Lambdaのコードを改良する	63
5.4. テストする	67
補足	68
Alexaの開発者アカウントに関する問題	68
リファレンス	75
Alexa の最新情報について	75
スキル開発に関するご質問	75

教材に関するご意見やご要望など	75
改定履歴（主な変更点）	76

はじめに

この資料は アマゾンジャパン合同会社 が主催する **Alexa ハンズオントレーニング** のために作成された演習教材です。

所定のレクチャーを受けたあと、この教材のステップに従って実習課題を進めていくことで、スキル開発に必要な基本的な知識とテクニックを習得できるようにデザインされています。

Alexa ハンズオントレーニング は東京、大阪など主要都市で定期的に開催されています。遠方にお住いの方や、まとまった時間が取れない方は、オンラインセミナー **Alexa道場** の収録動画をご覧ください。すでにトレーニングに参加された方がご覧になっても、より理解を深めることができるでしょう。



[Alexa道場ホームページ](#)

トレーニングの学習目標

このトレーニングを修了すると以下のことができるようになります。

- 簡単なスキルの作成手順を体験することで、スキル開発の基本的なステップを実演できるようになる。
- 簡単なサンプル発話を自分で設計することで、音声インターフェースデザインの基礎を習得することができる。
- ビルトインスロットタイプ及びカスタムスロットタイプを使ったカスタムスキルを自分で作成できるようになる。
- セッションアトリビュートを活用したマルチターンのスキルを作成できるようになる。
- エンドポイントの開発環境として AWS Lambda を使用するメリットを体験することができる。

課題実習の環境要件

この教材の課題を行うには、以下の環境が必要です。

- ・インターネットに接続されているパソコン (Windows, Mac, Linux)
- ・Firefox または Chrome ブラウザ



この教材ではAlexaシミュレーターを使ってスキルの動作テストを行います。Amazon EchoをはじめとするAlexa搭載デバイスはなくても構いません。



管理者権限で利用できるパソコンの利用を推奨します。所属する会社から貸与されているパソコンの場合、厳しいセキュリティ設定のため利用するツールがうまく動作しない場合があります。

ASK SDK for Node.js のバージョン

この教材では、**Alexa Skills Kit SDK for Node.js Version 2** を使用しています。

Alexa Skills Kit SDK は github.com/alexa からオープンソースとして配布され無料で利用できます。日本語ドキュメントはこちらからアクセスしてください。

[Alexa Skills Kit SDK for Node.js Version 2 日本語ドキュメント](#)

ダウンロード

最新版教材のダウンロード

Echoデバイス及びAlexaの進化に伴い、Alexaのスキル開発環境も凄まじいスピードで進化を遂げています。この資料も数週間に一回のペースでアップデートを繰り返しています。これからこの教材を使ってスキル開発の学習を始める場合は、下記のリンクから最新版のPDFをダウンロードしてお使いください。

- [最新版の教材ダウンロード](#)

サンプルコードのダウンロード

この教材では、プログラミング経験がなくてもスキル開発のステップを体験できるように、あらかじめ用意したサンプルコードを利用します。下記のリンクからサンプルコードをダウンロードしておいてください。

- [Alexa_SampleCode_Basic.zip](#)

サンプルコードはZIP形式の圧縮ファイルになっています。ファイルをダウンロードしたら解凍ツールで解凍してください。

ZIPファイルを解凍すると、SJISまたはUTF-8というフォルダがあります。サンプルコードを開いてコピー＆ペーストする場合は、お使いのパソコンまたはテキストエディタの環境に合わせてどちらかの文字コードのファイルを選択してください。

プログラミング経験について

先に述べたように、この教材はプログラミング経験がなくてもスキル開発を体験できようとしてデザインされています。しかし、今後独自のスキルを開発するには最低限のプログラミング言語の知識は避けては通れません。

もしこれからスキル開発を始めようとされる方は、以下のような書籍を使ってプログラミングの基礎を学習されることをおすすめします。

特にスキル開発においては、ブラウザの画面を制御するためのJavaScriptではなく、Node.jsについて詳しく解説した書籍を選択した方が良いでしょう。

- 初めてのJavaScript 第3版
- Node.js超入門
- Node.js入門～サーバーサイドJavaScriptを根本から理解する

1. 初めて作るカスタムスキル

この実習ではシンプルなカスタムスキル「コーヒーショップ」を作成します。現在のところスキル内で代金を支払う機能を設けることはできませんので、注文した商品を店頭にて作り置きを依頼するためのスキルであると想定して下さい。

1.1. この実習の学習目標

この実習で学んだテクニックを使うと次のようなことができるようになります。

- スキルビルダーを使って対話モデルを作成することができる。
- AWS LambdaとASK SDKを使ってエンドポイントのプログラムを作成できる。
- Alexa シミュレーターを使って作成したスキルをテストできる。

例 1. スキルの会話サンプル



「アレクサ、コーヒーショップを開いてコーヒーを注文して」



「コーヒーですね、ありがとうございます。今日は天気がいいので全部100円でいいですよ。またのご利用をお待ちしております。」



- 「呼び出し名」の意味と使い方を理解しよう
- 「インテント」と「サンプル発話」を適切に設計しよう

1.2. 対話モデルのデザイン

このセクションでは最初のステップ、対話モデルのデザインを行います。ユーザーがAlexaに対して何か話したら、Alexaがそれをどのように理解し、どのようなデータをエンドポイント（開発者のプログラムが動作するサーバー）に送るのかを設計します。



第3回 Alexa道場：音声インターフェイスをデザインしよう



このセクションでは、開発者コンソールを使用します。



参考ドキュメント

- カスタムスキルの構築手順

1.2.1. 開発者コンソールへのログイン

- Webブラウザ (Firefox または Chrome)で開発者コンソールへアクセスします。

<https://developer.amazon.com/ja/>

- 「Developer Console」のリンクをクリックします。

The screenshot shows the Amazon Developer Services and Technologies homepage. At the top, there's a navigation bar with 'Developer Console' highlighted in red. Below the header, there are several service sections:

- amazon alexa**: Described as "新しいテクノロジーによる自然で直感的なインターフェースの音声サービスを開発".
- amazon appstore**: Described as "Amazon Fire TV、Fireタブレット、モバイルプラットフォーム向けのAndroidアプリ・ゲームを開発".
- aws**: Described as "アマゾン ウェブ サービス 信頼性、拡張性、低コストを兼ね備えたクラウド コンピューティングサービス".
- amazon dash**: Described as "Dash Replenishment 自動的に再注文、快適なカスタマーエクスペリエンスを構築".

- ログインページが表示されたら、お手持ちのAmazon.co.jpアカウントでログインします。



- 初めて開発者アカウントを作成する場合でも、ログインするだけで自動的に登録画面に遷移します。Echoデバイスをお持ちの方は、そのデバイスをセットアップした時に使用したアカウントをお使いください。作成したスキルをお手持ちのデバイスでもテストできるので便利です。
- Amazon.co.jp(日本)とAmazon.com(米国)アカウントの両方をお持ちの方は、それぞれ異なるパスワードを設定し、Amazon.co.jpのアカウントでログインする必要があります。もし同じパスワードを設定している場合は、自動的にAmazon.comのアカウントでUSのスキルを開発することになりますのでご注意ください。
- よくわからない場合はサポートスタッフにお声がけください。

4. 初めてログインした場合、開発者アカウントの登録画面が表示されます。必要事項を入力してください。



この画面が表示されなかった場合、または英語のインターフェースで表示されてしまった場合は近くのサポートスタッフにお声がけください。

申請

1. プロフィール情報 2. App Distribution Agreement 3. 支払い

*は必須項目を示しています。

国/リージョン*

日本

名*

Tsuyoshi

姓*

Seino

Eメールアドレス*

cm-alexa-hanson@classmethod.jp

電話番号*

e.g. 212-555-1212, +44 0161 715 3369

11111111111111

Fax番号

開発者名*

Amazon.co.jpのアプリに表示されます

清野剛史

開発者名(ふりがな)*

開発者または会社名のふりがな

せーの つよし

開発者概要

最大文字数: 4000, 残り: 4000

住所1*

[REDACTED]

5. 利用規約に同意します。

申請

1. プロフィール情報 2. App Distribution Agreement 3. 支払い

English | 中文 (Chinese)* | 日本語 (Japanese)*

Last updated October 4, 2017

Current developers see what's changed.

APP DISTRIBUTION AND SERVICES AGREEMENT

This is an agreement between Amazon Digital Services LLC, Amazon Media EU S.a.r.l., Amazon Services International, Inc., Amazon Servicos de Varejo do Brasil Ltda., Amazon.com Int'l Sales, Inc., and Amazon Australia Services, Inc., and, if you are a resident of India that develops an Alexa Skill (as defined below), Amazon Seller Services Pvt Ltd (each, individually, an "Amazon Party" and, together with their affiliates, "Amazon," "we" or "us") and you (if registering as an individual) or the entity you represent (if registering as a business) ("Developer" or "you"). Any other Amazon affiliate that we designate is also an Amazon Party.

- Structure of Agreement.** This agreement (the "Agreement") includes the body of the agreement below, all schedules to this agreement ("Schedules"), and all terms, rules and policies that we make available for participating in this program, including on our developer portal (together, the "Program Policies"). However, the terms in each Schedule only apply to you if you engage in the activity or use the Program Materials (defined in Section 3) to which the Schedule applies (for instance, the terms of the Distribution Schedule only apply to you if you submit a product to us to sell, distribute, or promote). Please carefully read the Agreement before clicking to accept it.
- Our Program.** Our program (the "Program") allows end users to purchase, download, and access mobile and non-mobile software applications, games, and other digital products, and to use related services that we make available (for instance, Amazon GameCircle). "Apps" are software applications, games,

6. 収益化についての質問に答えて「保存して続行」ボタンをクリックしてください。今回は全て「いいえ」を選択します。

申請

1. プロフィール情報 2. App Distribution Agreement 3. 支払い

*は必須項目を示しています。

有料アプリや有料ゲーム、アプリ内課金やゲーム内課金、又はスキル利用による現金報酬の受取り等、収益化を検討されていますか？*

 いいえ
 はい

Amazon モバイル広告ネットワークや、モバイルアソシエイトからの広告をアプリで表示して、収益化する計画がおありますか？*

 いいえ
 はい

注意: 「いいえ」と答えた場合でも、後でアプリの収益化は可能です。

キャンセル

保存して続行

7. 開発者コンソールのTOP画面が表示されると開発者アカウントが正しく作成されています。

The screenshot shows the Amazon Developer Dashboard. At the top, there's a navigation bar with links for 'Dashboard', 'Apps & Services', 'Alexa', 'Software & Games', 'Login with Amazon', 'Dash Services', 'Report', and 'Settings'. On the far right of the header are three small circular icons labeled 'AO', '?', and a magnifying glass.

The main content area has two sections:

- 通知 (Notifications):** A table with two columns: 'All' and 'Critical'. The 'All' column has one entry: '通知はありません' (No notifications). The 'Critical' column is empty.
- アナウンス (Announcements):** A table with four columns: 'Title', 'Published Date', 'Content', and 'Last Updated'. The data is as follows:

Title	Published Date	Content	Last Updated
Alexa and Echo Devices Now Available to Customers in Italy and Spain	2018/10/24	Amazon Polly in Alexa Skills now Generally Available	2018/10/23
Alexa Skills Kit Expands to Include Canadian French	2018/10/10	Add Consumable In-Skill Purchasing To Your Skill	2018/09/25
New Alexa Presentation Language (Preview) for Building Voice-First Multimodal Experiences	2018/09/20	New Alexa Smart Home Development Tools	2018/09/20

1.2.2. スキルの新規作成

- 「Alexa」メニューから「Alexa Skills Kit」をクリックします。

The screenshot shows the Amazon Developer Dashboard with the 'Alexa Skills Kit' section highlighted by a red box. Below it, the 'Alexa Voice Service' section is visible. The '通知' (Notifications) and 'アナウンス' (Announcements) sections are also shown.

- 「スキル名」の入力とスキルのデフォルト言語を選択します。スキル名とは、このスキルを公開した際、スキルストアに表示されるスキルの名称です。ここではスキル名を「コーヒーショップ」、スキル作成時のデフォルトは「日本語」を選択します。
「スキルに追加するモデルの選択」では「カスタム」を選択し「スキルを作成」ボタンをクリックします。

The screenshot shows the 'Create New Skill' wizard in the Alexa Developer Console. The 'Skill Name' field contains 'コーヒーショップ'. The 'Default Language' dropdown is set to '日本語(日本)'. The 'Add Model' section has 'カスタム' selected. The 'Create Skill' button is highlighted with a red box.

- スキルビルダーの画面が表示されます。

右側のチェックリストは、この画面では4つの作業ステップあり、全てのステップをクリアすると、次のステップの「テスト」へ進めることを表しています。
チェックリストの手順に従い、「1.呼び出し名」のボックスをクリックします。



左側パネルの「カスタム」をクリックするといつでもこの画面に戻ることができます。

alexa developer console

日本語

カスタム

呼び出し名

インテント(4) ピックアップインテント(4)

AMAZON.CancelIntent
AMAZON.HelpIntent
AMAZON.StopIntent
AMAZON.NavigateHomeIntent

スロットタイプ(0) ピックアップ

JSONエディター

インターフェース
エンドポイント

開発を開始するには

スキルビルダーのチェックリスト

1. 呼び出し名 > カスタムの呼び出し名を入力します
2. インテント、サンプル、スコープ > 少なくとも1つのインテントと1つのサンプル発話を追加してください
3. モデルをビルト > 正常にモデルをビルトします
4. エンドポイント > ウェブサービスのエンドポイントを設定して、スキルのリクエストを処理します

4. スキルの呼び出し名を入力します。

このスキルを起動するための呼び出す名を設定します。呼び出し名は2~50文字である必要があります。ここでは「コーヒーショップ」と入力し「モデルを保存」をクリックします。



これで、ユーザーは「アレクサ、コーヒーショップを開いて、○○して。」のようにスキルを起動してスキルと会話できるようになります。

日本語

モデルを保存 モデルをビルト

カスタム

呼び出し名

インテント(3) ピックアップインテント(3)

AMAZON.CancelIntent
AMAZON.HelpIntent
AMAZON.StopIntent

スロット値(0) ピックアップ

JSONエディター

インターフェース
エンドポイント

アカウントリンク

アクセス権限

呼び出し名の要件

呼び出し名は2語以上でなければなりません。また、使用できるのはひらがな、カタカナ、漢字、小文字のアルファベット、スペースのみです。数字などは文字で表現しなければなりません。(例:「二十一」など)

呼び出し名には、「起動して」、「開いて」、「聞いて」、「教えて」、「読み込んで」、「開始して」、「有効にして」などの起動フレーズを使用することはできません。「アレクサ」、「アマゾン」、「エコー」、「コンピューター」などのウェイクワードや「スキル」、「アプリ」などの単語は使用できません。カスタムスキルの呼び出し名についての詳細は[こちら](#)。

スキルの対話モデルをビルトするまで、スキルの呼び出し名の変更は反映されません。ビルトを正常に完了させるには、スキルの対話モデルに少なくとも1つのサンプル発話をあるインテントが含まれている必要があります。カスタムスキルの対話モデルの作成についての詳細は[こちら](#)。



画面下の「呼び出し名の要件」は必ず読むようにしてください。使用できない文字や単語などの重要な情報が記載されています。

参考ドキュメント



- ユーザーによるカスタムスキルの呼び出し
- カスタムスキルの呼び出し名を決定する

5. 左パネルの「カスタム」をクリックし元の画面に戻ります。
6. スキルビルダーのチェックリストの「1.呼び出し名」に緑のチェックが付いていることを確認します。

The screenshot shows the Alexa Developer Console interface. On the left, there's a sidebar with tabs like 'スキル一覧', 'kopiershop', 'ビルド', 'テスト', '公開', '認定', and 'レポート'. The 'カスタム' tab is selected. Below it, there's a section for '対話モデル' (Dialog Model) with a dropdown for '日本語'. The main area shows a 'Skill Builder' interface with a large central window titled 'Alexa開発者コンソール: PetMatchIntent'. This window has sections for 'Interaction Model', 'Intents', and 'Sample Utterances'. To the right of this central window is a vertical sidebar with the title 'スキルビルダーのチェックリスト' (Skill Builder Checklist). The checklist consists of four items, each with a green checkmark:

- 1.呼び出し名 >** スキルの呼び出し名を入力します
- 2.インテント、サンプル、スロット >** 少なくとも1つのインテントと1つのサンプル発話を追加してください
- 3.モデルをビルド >** 正常に対話モデルをビルドします
- 4.エンドポイント >** ウェブサービスのエンドポイントを設定して、スキルのリクエストを処理します

1.2.3. 対話モデルの作成

対話モデルのデザインに取りかかります。ここでは、ユーザーがどのように発話すれば、どのようにAlexaがその意味を解釈し、その結果をどのようにイベントとしてエンドポイントのサーバーに伝えるべきかをデザインします。これらの入力データは「ビルド」という作業を経てAlexaの学習データとして取り込まれます。

1. スキルビルダーのチェックリストの「2.インテント、サンプル、スロット」のボックスをクリックします。



左パネルの「インテント」の右にある「追加」ボタンをクリックしても同じ画面に遷移します。

The screenshot shows the Alexa Developer Console interface. On the left, there's a sidebar with a language dropdown set to '日本語' (Japanese) and a 'カスタム' (Custom) section containing a '対話モデル' (Dialog Model) tab. Under '対話モデル', there are sections for '呼び出し名' (Invocation Name), 'インテント(4)' (Intents (4)), 'ビルトインインテント(4)' (Built-in Intents (4)), and 'スロットタイプ(0)' (Slot Type (0)). A red box highlights the '+ 追加' (Add) button next to the intent count. Below these are 'JSONエディター' (JSON Editor), 'インターフェース' (Interface), and 'エンドポイント' (Endpoint). In the center, there's a main panel titled '開発を開始するには' (Get Started) showing a preview of the Alexa developer console with an 'Intent / PetMatchIntent' tab. To the right, there's a large panel titled 'スキルビルダーのチェックリスト' (Skill Builder Checklist) with four items:

- 1.呼び出し名 > カスタムの呼び出し名を入力します (Completed)
- 2.インテント、サンプles、スロット > 少なくとも1つのインテントと1つのサンプル発話を追加してください (Incomplete, highlighted with a red box)
- 3.モデルをビルド > 正常に対話モデルをビルドします (Completed)
- 4.エンドポイント > ウェブサービスのエンドポイントを設定して、スキルのリクエストを処理します (Completed)

2. カスタムインテントの名前を入力します。ここでは **OrderIntent** と半角英文字で入力し「カスタムインテントを作成」ボタンをクリックします。

インテントを追加

インテントとは、ユーザーが話しかけたリクエストを実行するアクションのことです。インテントについての詳細は[こちら](#)。

- カスタムインテントを作成 [?](#)
- Alexaのビルトインライブラリから既存のインテントを使用 [?](#)

ビルトインインテントの使用についての詳細は[こちら](#)。

ビルトイン名	説明
標準 17 個のビルトイン	停止、キャンセル、ヘルプなど一般的な操作のインテントです。



OrderIntent のスペルは間違えないよう正確に入力してください。

- インテント OrderIntent に紐づくサンプル発話を入力します。サンプル発話とは、インテントを呼び出すためにユーザーがAlexaに話しかけるフレーズのことです。ここでは、コーヒーショップのスキルを使ってコーヒーを注文するためのフレーズ「コーヒーを注文して」と入力します。

インテント / OrderIntent

サンプル発話(1) [?](#)

このインテントの呼び出しに使われると考えられる発話	+
コーヒーを注文して	-

インテントスロット(0) [?](#)

順序 ?	名前 ?	スロットタイプ ?	アクション
1	新しいスロットを作成	+ スロットタイプを選択	ダイアログを編集 削除

4. インテントのサンプル発話の入力が完了したら「モデルを保存」をクリックし、最後に「モデルをビルト」ボタンをクリックします。モデルのビルトには1~2分かかる場合があります。

順序	名前	スロットタイプ	アクション
1	新しいスロットを作成	+ スロットタイプを選択	ダイアログを編集 削除

5. モデルのビルトが完了したら対話モデルの構築は完了です。左側の「カスタム」タブをクリックします。「ビルト」のトップ画面に戻り、スキルビルダーのチェックリストを確認しましょう。「3. モデルをビルト」まで完了し緑のチェックマークがついていればOKです。

- 1.呼び出し名 > スキルの呼び出し名を入力します ✓
- 2.インテント、サンプル、スロット > 少なくとも1つのインテントと1つのサンプル発話を追加してください ✓
- 3.モデルをビルト >** 正常に対話モデルをビルトします ✓
- 4.エンドポイント > ウェブサービスのエンドポイントを設定して、スキルのリクエストを処理します ✓

6. 「4. エンドポイント」のボックスをクリックします。

The screenshot shows the Alexa Developer Console interface. On the left, there's a sidebar with language selection (日本語), a custom model section, and a list of intents (OrderIntent, AMAZON.CancelIntent, AMAZON.HelpIntent, AMAZON.StopIntent, AMAZON.NavigateHomeIntent) and slot types. Below that are sections for JSON Editor, Interfaces, and Endpoints. On the right, there's a large panel titled "開発を開始するには" (Getting Started) showing a screenshot of the developer tools. To the right of that is a section titled "スキルビルダーのチェックリスト" (Skill Builder Checklist) with four items: 1.呼び出し名 (checked), 2.インテント、サンプル、スロット (checked), 3.モデルをビルド (checked), and 4.エンドポイント (unchecked). Item 4 is highlighted with a red border.

7. サービスのエンドポイントの種類では「AWS LambdaのARN」を選択します。

The screenshot shows the "Endpoint" configuration screen. It includes a note about how endpoints handle POST requests and a list of service endpoint types. The "AWS LambdaのARN (推奨)" option is selected and highlighted with a red border.

8. 「クリップボードにコピー」のリンクをクリックし、スキルIDの文字列をコピーしておきます。この後の処理で必要となりますので、テキストエディタ等を開いてこの文字列を保存しておいてください。

エンドポイント



ユーザーがAlexaスキルと対話すると、エンドポイントはPOSTリクエストを受け取ります。このリクエスト本文には、サービスがロジックを実行してJSON形式の応答を生成できるパラメーターが含まれています。AWS Lambdaのエンドポイントについて詳しくは、[こちらをご覧ください](#)。サービスがこちらに記載されている要件を満たしていれば、ユーザー定義のHTTPS Webサービスをホスティングできます。

サービスのエンドポイントの種類

スキルのサービスエンドポイントをホスティングする方法を選択します。

AWS LambdaのARN (推奨)

スキルID (?)

amzn1.ask.skill.ebc725a3-3192-486d-b1a3-9d6c24142ac1

クリップボードにコピー

デフォルトの地域
(必須)

arn:aws:lambda:<location>:<aws_account_id>:function:<lam

北米 (?)
(オプション)

arn:aws:lambda:us-east-1:<aws_account_id>:function:<lamb

以上で、対話モデルの作業は終了です。ここまでで、Alexaはユーザーの「コーヒーを注文して」という要求フレーズを理解し、エンドポイントのプログラムに **OrderIntent** というリクエストを送信できるようになります。

次に、リクエスト **OrderIntent** を受け取る側、つまりエンドポイントのプログラムを書くステップに移ります。

1.3. エンドポイントの開発



OrderIntentを受け取り、何らかの処理をした後、Alexaに応答を返すイベント駆動のサーバープログラムを作成します。サーバーのプラットフォームには、AWSのLambdaというサービスを利用します。スキルのプログラムコードは、Alexaスキル開発用のライブラリ **Alexa Skills Kit SDK for Node.js** を使って作成します。



このセクションでは、AWSマネージドコンソールを使用します。

参考ドキュメント



- カスタムスキルのAWS Lambda関数を作成する
- Alexaから送信されたリクエストを処理する

1.3.1. AWSコンソールへのログイン

- 以下のリンクからAWSポータルにアクセスします。

URL: <https://aws.amazon.com/jp/>

- 「コンソールへログイン」をクリックします。



- AWSアカウントIDを入力します。



AWSアカウントは無料で作成できますが、クレジットカードの登録が必要になります。どうしてもアカウントの作成が難しい場合はサポートスタッフにご相談ください。



4. パスワードを入力し、ログインします。



本来はセキュリティ上の理由からルートユーザーでのログインは推奨されません。通常はIAMで開発用ユーザーを追加し、開発用ユーザーでログインし直して作業を行ってください。



5. Lambdaを選択してクリックします。Lambdaが表示されていない場合は、上の検索ボックスに Lambda と入力すると表示されます。Lambdaはコンピューティングのカテゴリにあります。

AWS サービス

サービスを名前、あるいは機能で検索(例: EC2、S3、VM、ストレージ)。

最近アクセスしたサービス

- S3
- IAM
- Amazon Polly
- CloudFormation
- Lambda**

すべてのサービス

ソリューションの構築

シンプルなウィザードと自動化されたワークフローで作業を開始します。

- 右上のリージョンを「アジアパシフィック（東京）」に変更し、「関数の作成」ボタンをクリックしてください。



2018年10月時点でAlexaに接続できるリージョンは、「米国東部(バージニア北部)」「米国西部(オレゴン)」「EU(アイルランド)」「アジアパシフィック(東京)」の4つとなっています。これ以外のリージョンでは、Alexaに接続できなくなるので注意してください。

AWS Lambda > 関数

関数 (0)

アクション 関数の作成

タグや属性によるフィルター、またはキーワードによる検索

関数名	説明	ランタイム	コードサイズ	最終更新日時
表示するデータがありません。				

- 「一から作成」を選択し、名前欄に「**CoffeeShop**」と入力します。ランタイムのバージョンを「**Node.js 8.10**」に変更し、「ロール」のプルダウンメニューから「テンプレートから新しいロールの作成」を選択してください。ロール名には適当な名前を付けてください。（例：AlexaRoleなど）

関数の作成

一から作成

名前: coffeeshop

ランタイム: Node.js 8.10

ロール: テンプレートから新しいロールを作成

この新しいロールの範囲は現在の関数になります。このロールを他の関数で使用する場合は、IAMコンソールで変更できます。

8. ポリシーテンプレートは、「シンプルなマイクロサービスのアクセス権限」を選択してください。



今回は Amazon CloudWatch Logs と Amazon DynamoDB にアクセスできるシンプルなロールテンプレートを選択します。S3など他のサービスにアクセスしたい場合は、適切なロールを追加してください。

から作成

CloudFormation のスタック読み取り専用アクセス権限

AMI の読み取り専用アクセス権限

S3 オブジェクトの読み取り専用アクセス権限

Elasticsearch のアクセス権限

SES バウンスのアクセス権限

テストハーネスのアクセス権限

シンプルなマイクロサービスのアクセス権限

VPN 接続のモニタリングのアクセス権限

SQS ポーリングのアクセス権限

AWS IoT ボタンのアクセス権限

Amazon Rekognition データなしアクセス権限

Amazon Rekognition 読み取り専用アクセス権限

Amazon Rekognition 書き込み専用アクセス権限

AWS Config ルールのアクセス権限

AWS Batch アクセス権限

SNS 発行ポリシー

KMS の復号化アクセス権限

Basic Edge Lambda アクセス権限

9. 下図のようになっていればOKです。画面下にある「関数の作成」ボタンをクリックしてください。

一から作成 情報

名前
CoffeeShop

ランタイム
Node.js 8.10

ロール
関数のアクセス許可を定義します。新しいロールは、作成後の数分間は使用できないことがあります。Lambda 実行ロールの詳細については、[こちら](#)を参照してください。

テンプレートから新しいロールを作成
Lambda は、選択したポリシーテンプレートのアクセス許可を持つロールを自動的に作成します。基本的な Lambda のアクセス権限 (CloudWatch へのログ) が自動的に追加されることに注意してください。関数が VPC にアクセスする場合、必要なアクセス権限も追加されます。

ロール名
新しいロールの名前を入力します。
AlexaRole

① この新しいロールの範囲は現在の関数になります。このロールを他の関数で使用する場合は、IAM コンソールで変更できます。

ポリシーテンプレート
1つ以上のポリシーテンプレートを選択します。関数が作成される前にロールが生成されます。各ポリシーテンプレートがロールに追加するアクセス権限については、[こちら](#)を参照してください。

シンプルなマイクロサービスのアクセス権限 X

[キャンセル](#) [関数の作成](#)

10. 関数が作成されると次のような画面になります。

Lambda > 関数 > CoffeeShop

function: CoffeeShop ARN - arn:aws:lambda:ap-northeast-1:4066-5023-7351

スロットリング デfault 条件 デfault アクション デfault テストイベントの選択 テスト 保存

おめでとうございます。Lambda 関数「CoffeeShop」が正常に作成されました。これでコードおよび設定を変更できるようになりました。関数をテストする準備ができたら、「テスト」ボタンをクリックしてテストイベントを入力してください。

設定 モニタリング

▼ Designer

トリガーの追加 下のリストのトリガーをクリックして、トリガーを関数に追加します。

- API Gateway
- AWS IoT
- Alexa Skills Kit
- Alexa Smart Home
- CloudWatch Events

CoffeeShop

Amazon CloudWatch Logs

Amazon DynamoDB

関数のロールでアクセスできるリソースが、ここに表示されます

関数コード 情報

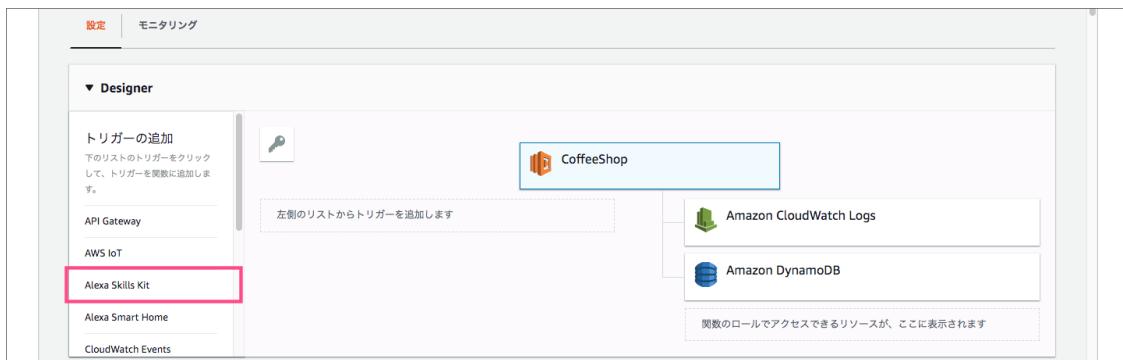
フィードバック 日本語

© 2008-2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. プライバシーポリシー 利用規約

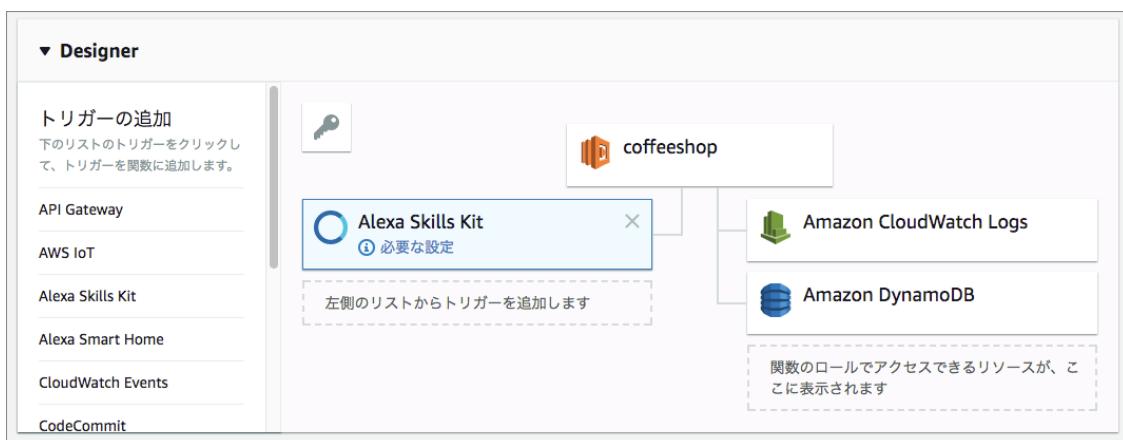
1.3.2. トリガーを追加する

Lambda関数は様々なサービスと接続し呼びだすことができます。トリガーとは、作成した関数が何によって呼び出されるかを指定します。ここでは **Alexa Skills Kit** を指定します。

1. 画面左「トリガーの追加」から **Alexa Skills Kit** をクリックします。



2. Alexa Skills Kit が追加されると下のような表示になります。



3. トリガーの設定では、スキルIDを検証を「有効」に設定し、アプリケーションIDのテキスト入力フィールドに、[対話モデルの作成]のステップ7でコピーしておいたスキルIDを貼り付け、「追加」ボタンをクリックしてください。



スキルIDの検証を「有効」にすることで、このLambda関数が、意図しない他のAlexaスキルから呼び出されないよう制限することができます。

トリガーの設定

① スキル ID 検証は、スキルからの受信リクエストでスキル ID を検証するための簡単な方法です。これを設定するには、Alexa Skills Kit ダッシュボードにあるスキルのスキル ID (アプリケーション ID とも呼ばれます) を入力します。 詳細は[こちら](#)

スキル ID 検証
 有効 (推奨)
 無効

アプリケーション ID

Lambda は、このトリガーから Lambda 関数を呼び出すのに必要な Amazon Alexa のアクセス許可を追加します。Lambda アクセス許可モデルの詳細については[こちら](#)を参照してください。

キャンセル 追加

4. 「保存」をクリックしてください。

coffeeshop 限定条件 ▾ アクション ▾ テストイベントの選択.. ▾ テスト 保存

おめでとうございます。Lambda 関数「coffeeshop」が正常に作成されました。これでコードおよび設定を変更できるようになりました。関数をテストする準備ができたら、「テスト」ボタンをクリックしてテストイベントを入力してください。

設定 モニタリング

▼ Designer

トリガーの追加 下のリストのトリガーをクリックして、トリガーを関数に追加します。

API Gateway
AWS IoT
Alexa Skills Kit

The Designer panel shows the 'coffeeshop' Lambda function connected to three resources: Alexa Skills Kit (with a note '① 保存されていない変更'), Amazon CloudWatch Logs, and Amazon DynamoDB.

1.3.3. Lambda関数のコードをアップロードする

1. Designer パネルの中に表示されている「CoffeeShop」と書かれたLambdaのアイコンをクリックしてください。

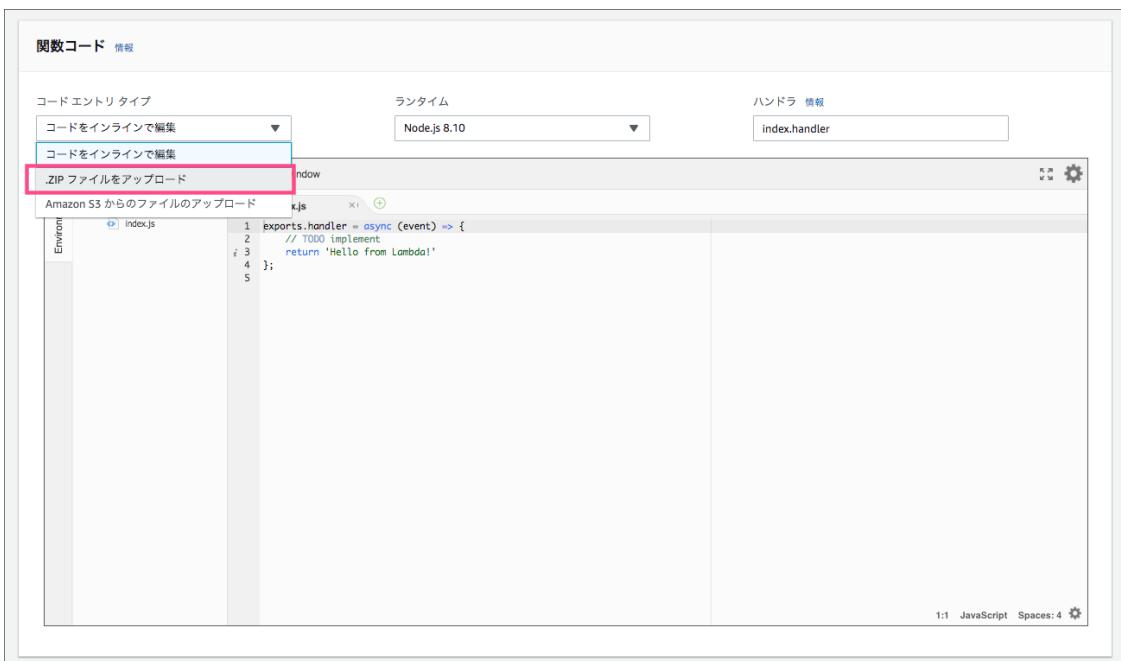
▼ Designer

トリガーの追加 下のリストのトリガーをクリックして、トリガーを関数に追加します。

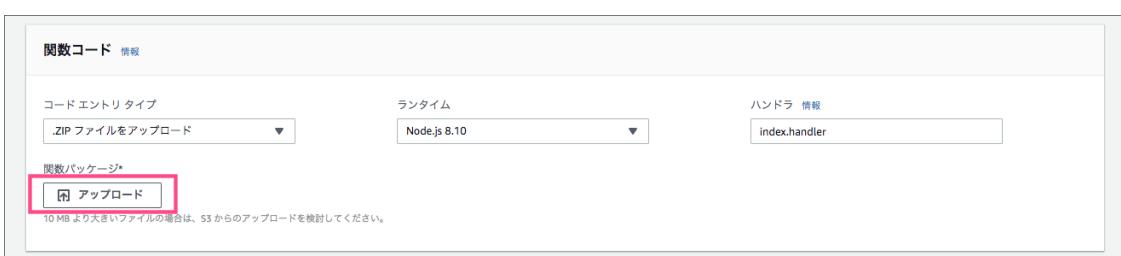
API Gateway
AWS IoT
Alexa Skills Kit
Alexa Smart Home
CloudWatch Events

The Designer panel shows the 'CoffeeShop' Lambda function selected for upload. A note indicates '必要な設定' (Required settings).

2. 画面の下方に関数のコードエディタが表示されます。左上のコードエントリタイプのプルダウンメニューから「.ZIPファイルをアップロード」を選択してください。



3. 「アップロード」ボタンをクリックしてください。



4. 解凍したサンプルファイルのフォルダの中から、**Coffeeshop.zip** ファイルを探し選択してください。



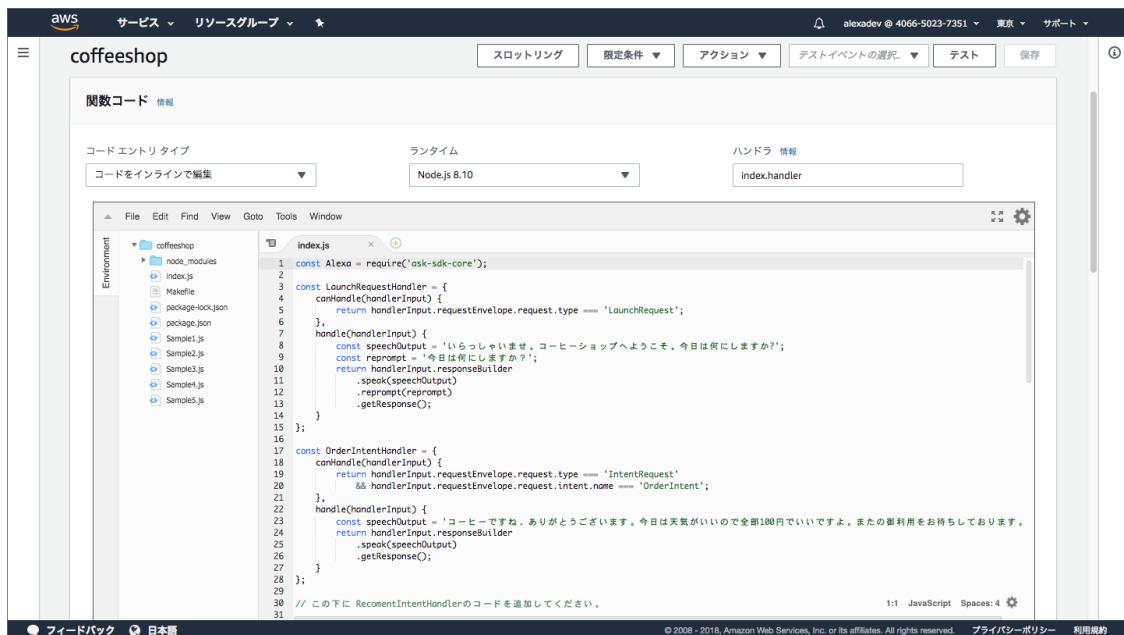
ソースコードとライブラリを含めたZIPファイルの作り方は、以下のSK SDK v2 for Node.js のドキュメントを参照してください。
[Setting Up The ASK SDK](#)

5. 画面右上の「保存」ボタンをクリックします。

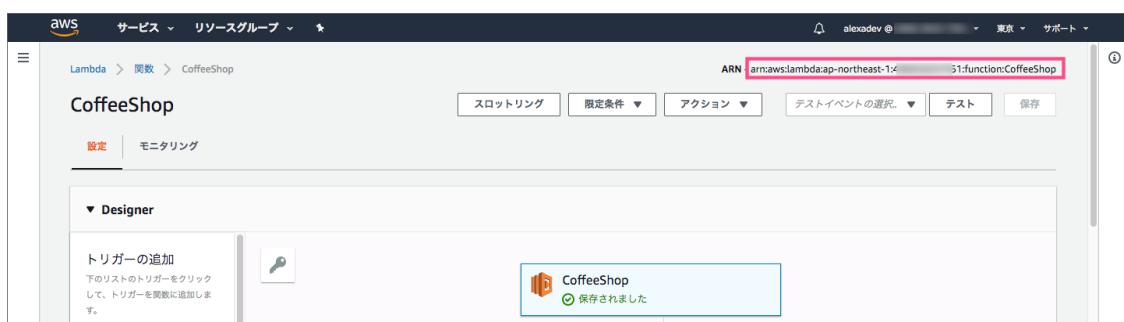


6. 課題1のサンプルコードが読み込まれた状態でコードエディタが開きます。ここでは

デフォルトで呼び出されるコード **index.js** と ASK-SDK のライブラリ **ask-sdk-core** および **ask-sdk-model** もデプロイされていることが確認できます。



7. 画面右上に表示されている、ARNをクリップボードにコピーしておきます。



コピーする文字列は、以下ののような形式になります。

```
arn:aws:lambda:ap-northeast-1:XXXXXXX:function:coffeeshop
```

これが、Alexa からLambda関数を呼び出すエンドポイントのアドレスになります。この文字列が間違っていると目的のエンドポイントが見つからない、または別のエンドポイントに接続され、エラーとなるので注意してください。

次のセクションでは、ステップ(1)で作成した対話モデルに、このセクションで作成したエンドポイントのLambda関数を登録します。

1.4. エンドポイントを登録する



対話モデルに、Lambda関数で作ったエンドポイントを登録し、Alexaから適切にLambda関数を呼び出せるようにします。



このセクションでは開発者コンソールを使用します。

- スキルビルダーの「カスタム」の画面に戻り、スキルビルダーのチェックリストから「4.エンドポイント」をクリックします。



左パネルの「エンドポイント」をクリックしても同じです。

The screenshot shows the Alexa developer console interface. On the left, there's a sidebar with sections like '日本語', 'カスタム', '対話モデル', and 'エンドポイント' (which is highlighted with a pink border). The main content area has two panels: '開発を開始するには' (with a video thumbnail) and 'スキルビルダーのチェックリスト'. The checklist on the right lists several steps, with step 4 ('エンドポイント') highlighted with a red border.

ステップ	説明	状況
1.呼び出し名	スキルの呼び出し名を入力します	完了
2.インテント、サンプル、スロット	少なくとも1つのインテントと1つのサンプル発話を追加してください	完了
3.モデルをビルド	正常に対話モデルをビルドします	完了
4.エンドポイント	ウェブサービスのエンドポイントを設定して、スキルのリクエストを処理します	未実行

- サービスのエンドポイントをホスティングする方法は「AWS LambdaのARN」を選択します。「デフォルトの地域」のテキストボックスに先ほどコピーしておいたLambda関数のARNを貼り付けてください。その他の項目はデフォルトのままにしてください。

日本語 エンドポイントを保存

カスタム

対話モデル

呼び出し名

- インテント(5) **+ 追加**
 - OrderIntent
- ビルトインインテント(4)
 - AMAZON.CancelIntent
 - AMAZON.HelpIntent
 - AMAZON.StopIntent
 - AMAZON.NavigateHomeIntent

スロットタイプ(0) **+ 追加**

JSONエディター

エンドポイント

インターフェース

インテント履歴

エンドポイント

サービスのエンドポイントの種類

スキルのサービスエンドポイントをホスティングする方法を選択します。

AWS LambdaのARN **(推奨)** **スキルID** **amzn1.ask.skill:ebc725a3-3192-486d-b1a3-9d6c24142ac1** **クリップボードにコピー**

デフォルトの地域 **(必須)** **arn:aws:lambda:ap-northeast-1:406650237351:function:Cof**

北米 **(オプション)** **arn:aws:lambda:us-east-1:<aws_account_id>:function:<lambda_function_name>**

3. 「エンドポイントの保存」ボタンをクリックします。

日本語 エンドポイントを保存

カスタム

対話モデル

呼び出し名

- インテント(5) **+ 追加**
 - OrderIntent
- ビルトインインテント(4)

エンドポイント

サービスのエンドポイントの種類

スキルのサービスエンドポイントをホスティングする方法を選択します。

AWS LambdaのARN **(推奨)** **スキルID** **amzn1.ask.skill:ebc725a3-3192-486d-b1a3-9d6c24142ac1** **クリップボードにコピー**

デフォルトの地域 **(必須)** **arn:aws:lambda:ap-northeast-1:406650237351:function:Cof**

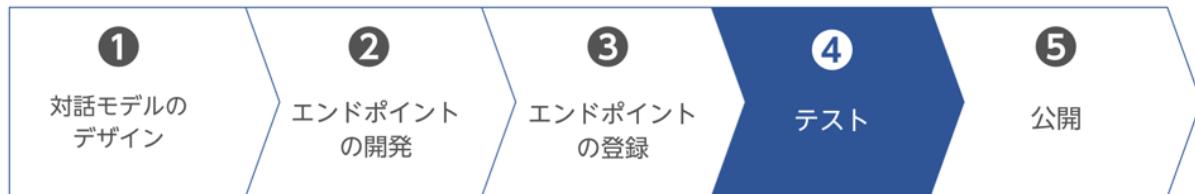
北米 **(オプション)** **arn:aws:lambda:us-east-1:<aws_account_id>:function:<lambda_function_name>**

エンドポイント登録が完了しました。

以上で、音声ユーザーインターフェースにエンドポイントを登録する作業が完了しました。
簡単ですね？

これでスキルはほぼ完成に近づきました。次のステップでは、あなたのスキルが正しく動作するかテストしてみましょう。

1.5. テストする



この段階で、あなたのスキルはほぼ出来上がっています。正しく動作するかどうかテストしてみましょう。もしうまく動作しなかった場合は、どこかの設定が間違っているのかもしれません。ステップ(1)から(3)に戻り、再確認しましょう。



このセクションでは開発者コンソールを使用します。

参考ドキュメント



- [スキルのテスト](#)
- [ユーザーによるカスタムスキルの呼び出し](#)

1.5.1. Alexaシミュレータでテストする

1. 「テスト」タブを開きます。テストが無効になっている場合はスイッチをクリックしてテストを有効に変更します。

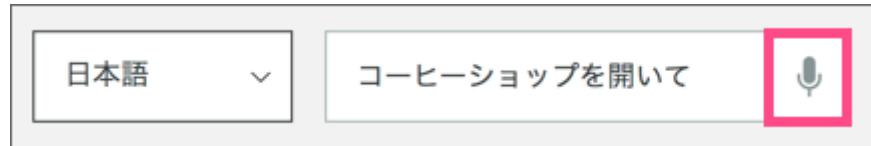


2. Alexaシミュレータを使って、スキルのテストをしましょう。言語が日本語になっていることを確認し、その隣のテキストボックスにAlexaに話しかけるフレーズをテキストで入力し[Enter]キーを押します。

リスト 1. 入力例（ウェイクワードは省略することができます）

コーヒーショップを開いてコーヒーを注文して

もしくは下図のマイクのアイコンをクリックした状態で、同様のフレーズをパソコンのマイクに向かって声で話しかけます。



3. 画面左側のパネルには、Alexaに送ったユーザーの発話と、Alexaからの応答メッセージが会話形式で表示されます。右側のパネルには、Alexaからスキルに送信されたJSONデータ (JSON入力) と、スキルからAlexaに送信されたJSONデータ (JSON出力) の中身を確認することができます。

alex developer console

日本語

日本語

JSON入力

```

1 - {
2   "version": "1.0",
3   "session": {
4     "new": true,
5     "sessionId": "amzn1.echo-api.session",
6     "application": {
7       "applicationId": "amzn1.ask.skill"
8     },
9     "user": {
10       "userId": "amzn1.ask.account.A1"
11     }
12   },
13   "context": {
14     "System": {
15       "application": {
16         "applicationId": "amzn1.ask.skill"
17       },
18       "user": {
19         "userId": "amzn1.ask.account.A1"
20       },
21       "device": {
22         "deviceId": "amzn1.ask.dev"
23       },
24     }
25   }

```

信されたJSONデータ

```

1 - {
2   "body": {
3     "version": "1.0",
4     "response": {
5       "outputSpeech": {
6         "type": "SSML",
7         "ssml": "<speak>コーヒーですね、ありがとうございます。今日は天気がいいので全部100円でいいですよ。またの御利
用をお待ちしております。
8       


JSON出力



エンドポイントからalexaに送  
信されたJSONデータ


```

4. JSON入力の中から、どのようなインテントが送られてきているかを確認しましょう。
5. JSON出力の中から `"outputSpeech": {"ssml": <speech> ...}` の文字列を探してみてください。レスポンスのJSONデータの中にAlexaが読み上げるテキストが埋め込まれていることがわかります。上部の右側の再生ボタンをクリックすると、Alexaが読み上げる音声を何度も再生することができます。

1.5.2. (オプション) Echoデバイスでテストする。

お手持ちのEchoデバイスを使ってテストする手順を説明します。はじめに、Alexaアプリのスキル一覧ページに開発中のスキルが表示されているかどうかを確認します。



ここではAlexaアプリを使用します。(<https://alexa.amazon.co.jp>)



まだ1台も自分のアカウントでEchoデバイスをセットアップしたことがない場合、Alexaアプリにログインするとデバイスのセットアップ画面が表示され、自分のスキルを確認することはできません。

1. Alexaアプリにログインし「スキル」タブをクリックします。

The screenshot shows the Alexa Skills Kit interface. On the left, a sidebar menu has 'Skills' highlighted with a pink rectangle. The main content area displays a skill named 'Hello World'. It includes a greeting message 'Hello' and a note: 'Welcome to the Alexa Skills Kit, you can say hello!'. There are also buttons for '続きを表示' (Show More) and '元に戻す' (Reset). At the top right, there's a link '詳細はこちら' (More details) and a back arrow.

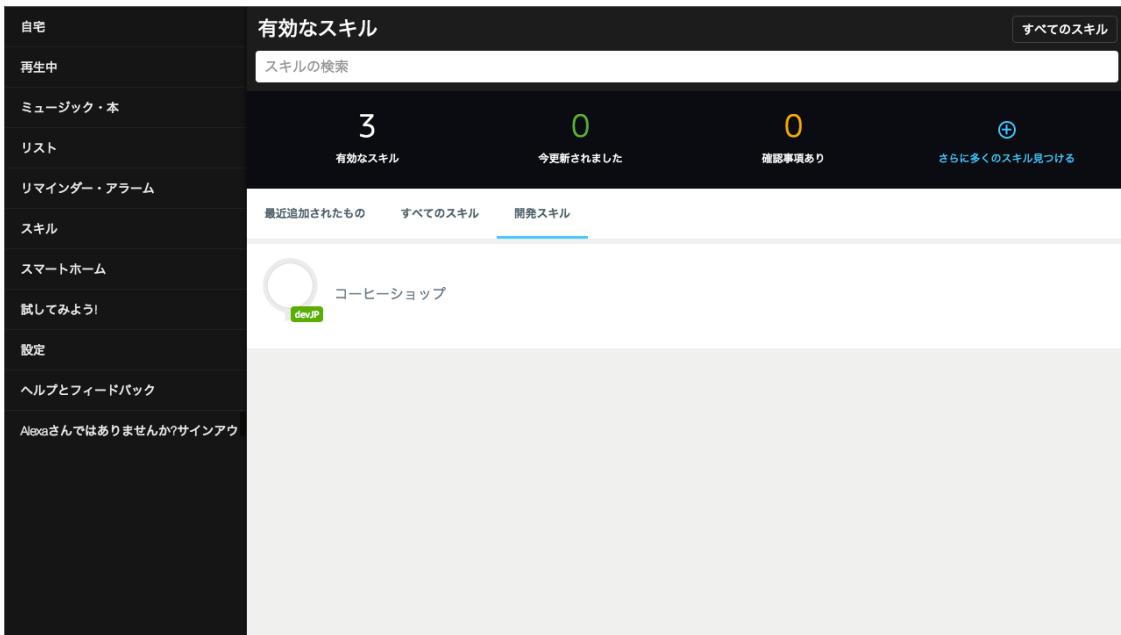
2. 「有効なスキル」をクリックします。

The screenshot shows the '有用なスキル' (Useful Skills) tab selected on the Alexa Skills page. The main content area displays several skill cards, including 'ハロウィンナイト', 'ハロウィンの練習', 'オバケ退治', '魔法使いの冒険', '忘れ物チェック', 'エスケープルーム', '毎日カニササレ', and 'あなたのダービーを聞かせよう'. Each card includes a thumbnail, the skill name, a star rating, and a brief description.

3. 「開発スキル」のタブをクリックします。

The screenshot shows the '開発スキル' (Development Skills) tab selected on the Alexa Skills page. The interface displays statistics: 3 useful skills, 0 updated skills, 0 verified items, and a plus sign for more skills. Below this, a list of recently added skills is shown, including 'NHKニュース' and '天気予報', both of which have a green 'devJP' icon indicating they are currently being developed.

現在開発中（非公開）のスキルのリストが表示されます。リストの中から、あなたが作成した「コーヒーショップ」スキルを探してください。アイコンの右下に緑色の**devJP** というマークのついたものが開発中のスキルを表しています。



4. スキルが有効になっていれば、開発者アカウントと同じアカウントでセットアップされているAlexa対応デバイス（Amazon Echoなど）でもテストすることができます。



5. もし、手元にAlexa対応デバイスがある場合は、それを使って動作テストをしてみてください。スキルを起動する際は、スキルの呼び出し名を忘れずに！

リスト 2. 呼び出しフレーズ

アレクサ、コーヒーショップを開いてコーヒーを注文して」



スキル開発の基本ステップは以上です。初めて作成したスキルは動きましたか？

1.5.3. うまく動かない場合

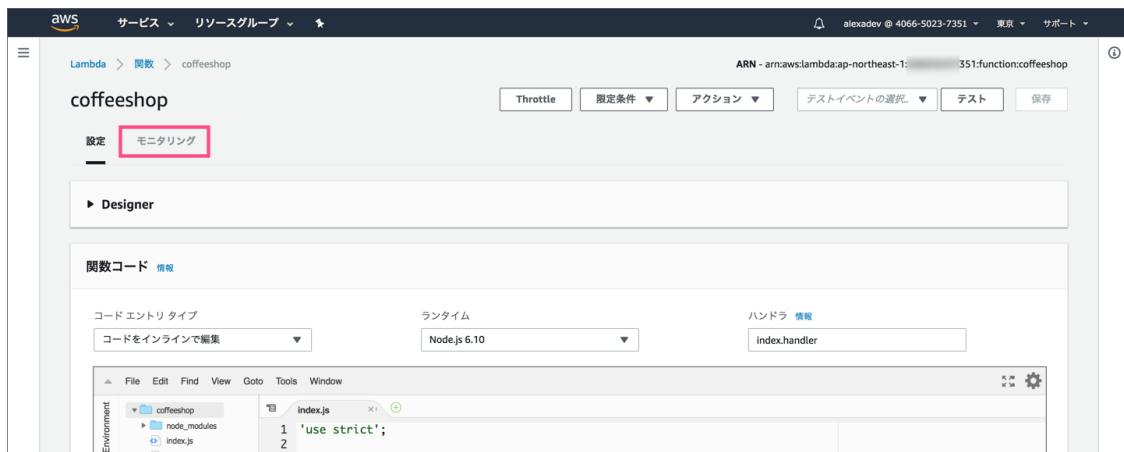
うまく動かない場合、様々な要因がありますが特に以下の設定を見直してみましょう。

- スキルビルダーのチェックリストは全て緑のアイコンに変わっていますか？
- スキルビルダー側に登録したエンドポイントのARNは正しいですか？
- スキルビルダーで追加したインテント名のスペルと、Lambda関数で登録されているインテント名のスペルは一致していますか？
- Alexaシミュレーターで、音声を使ってスキルを起動した際の「呼び出し名」の認識文字列と、スキルビルダーで設定した「呼び出し名」は完全に一致していますか？

1.5.4. 「スキルからの応答に問題があります」？

Alexaが上記のフレーズを言って期待する動作をしない場合、Lambdaのコード内で何らかのエラーが発生し正しくレスポンスを返していない状態です。Lambda側で何が起きているのか CloudWatchLog を使ってエラーシューティングを行ってみましょう。

1. Lambda関数の編集画面を開き、「モニタリング」をクリックします。



2. CloudWatchメトリクスの画面が開きます。「呼び出しカウント」の枠内にある「ログにジャンプ」をクリックします。

CloudWatch メトリクスの概要 (aggregated per hour)

呼び出しカウント 過去 24 時間 ▾ ログにジャンプ

呼び出し所要時間 過去 24 時間 ▾ ログにジャンプ

呼び出しエラー 過去 24 時間 ▾ ログにジャンプ

CloudWatch ログ

CloudWatch Metrics

CloudWatch Metrics

CloudWatch Metrics

- CloudWatchのログ画面が開きます。下に行くほど新しいログが表示されます。何らかのエラーが起こっている行を探します。フィルター機能を使って **Error** 等の文字列で検索しても良いでしょう。

CloudWatch ログ

すべて展開 行 テキスト

Error

時間 (UTC +0:00) メッセージ

2018-04-11T13:27:51 2018-04-11T13:27:51.066Z 2207eac2-3d8c-11e8-9114-65a3f19e8432 Unexpected exception: Error: No 'Uncaught' function defined

2018-04-11T13:27:51 2018-04-11T13:27:51.066Z 2207eac2-3d8c-11e8-9114-65a3f19e8432 ["errorMessage": "No 'Uncaught' function defined for event: Uncaught", "stackTrace": [

2018-04-11T13:29:06 4172 4ee4717c-3d8c-11e8-b59d-7b23248fd28 Unexpected exception: TypeError: Cannot read property 'value' of undefined: "e"

2018-04-11T13:29:06 4382 4ee4717c-3d8c-11e8-b59d-7b23248fd28 ["errorMessage": "Cannot read property 'value' of undefined: "e"

2018-04-11T13:29:06 6572 4f12fc46-3d8c-11e8-99c3-7537508a83c Unexpected exception: Error: No 'Uncaught' function defined for event: Uncaught

2018-04-11T13:29:06 6572 4f12fc46-3d8c-11e8-99c3-7537508a83c ["errorMessage": "No 'Uncaught' function defined for event: Uncaught", "stackTrace": [

2018-04-11T13:34:12 13:34:12 05a0fe11-3d8d-11e8-a3ab-67ba3fb4bb Unexpected exception: TypeError: Cannot read property 'value' of undefined: "e"

2018-04-11T13:34:12 13:34:12 05a0fe11-3d8d-11e8-a3ab-67ba3fb4bb ["errorMessage": "Cannot read property 'value' of undefined: "e"

2018-04-11T13:36:38 13:36:38 1992 5c310e8-3d8d-11e8-8d7-4d4c2b06e1b Unexpected exception: TypeError: Cannot read property 'value' of undefined: "e"

2018-04-11T13:36:38 13:36:38 1992 5c310e8-3d8d-11e8-8d7-4d4c2b06e1b ["errorMessage": "Cannot read property 'value' of undefined: "e"

2018-04-11T13:40:51 13:40:51 05a0fe11-3d8d-11e8-86e1-9dc42c167040 Unexpected exception: TypeError: Cannot read property 'value' of undefined: "e"

2018-04-11T13:40:51 13:40:51 05a0fe11-3d8d-11e8-86e1-9dc42c167040 ["errorMessage": "Cannot read property 'value' of undefined: "e"

2018-04-11T13:43:57 0402 62443e8f-3d8d-11e8-8d7-4d4c2b06e1b Unexpected exception: TypeError: Cannot read property 'value' of undefined: "e"

2018-04-11T13:43:57 0402 62443e8f-3d8d-11e8-8d7-4d4c2b06e1b ["errorMessage": "Cannot read property 'value' of undefined: "e"]

2018-04-11T13:43:58 0182 62443e8f-3d8d-11e8-8d7-4d4c2b06e1b {

4. エラーを見つけたら、メッセージを読みLambda関数内のエラー箇所を修正します。

5. 修正したら再びAlexaシミュレーターでテストをしましょう。正しく動作するまでこのデバッグ作業を繰り返します。根気よく頑張りましょう！

2. インテントの追加

2.1. この実習の学習目標

この実習では、ユーザーがコーヒーの注文とは関係のない発話をしてきた場合の受け取り方法と、ビルトイン・インテントの実装方法を学びます。

この実習で学んだテクニックを使うと次のようなことができるようになります。

- ・ インテントとは何かを説明できるようになる。
- ・ 任意のインテントと、インテントハンドラーを追加することができる。
- ・ ビルトイン・インテントの追加と実装方法を実演できるようになる。

2.2. 対話モデルの編集

例 2. スキルの会話サンプル



「アレクサ、コーヒーショップを開いて、おすすめは何？」



「今日は甘さスッキリのカフェラテがおすすめです。」



参考ドキュメント

- ・ インテント、発話、スロットの作成

2.2.1. インテントを追加する

1. コーヒーショップスキルのスキルビルダーの画面を開きます。
2. 左パネルにあるインテントの「追加」ボタンをクリックします。

開発を開始するには

スキルビルダーのチェックリスト

- 呼び出し名 > カスタムの呼び出し名を入力します (Green checkmark)
- インテント、サンプル、スロット > 少なくとも1つのインテントと1つのサンプル発話を追加してください (Green checkmark)
- モデルをビルト > 正常に対話モデルをビルトします (Green checkmark)
- エンドポイント > ウェブサービスのエンドポイントを設定して、スキルのリクエストを処理します (Green checkmark)

3. ユーザーの「おすすめは何？」という発話に対応するカスタムインテントを作成します。インテント名の入力フィールドに「RecommendIntent」と入力し、「カスタムインテントの作成」ボタンをクリックします。

インテントを追加

インテントとは、ユーザーが話しかけたリクエストを実行するアクションのことです。インテントについての詳細は[こちら](#)。

カスタムインテントを作成 [?](#)

Alexaのビルトインライブラリから既存のインテントを使用 [?](#)

4. インテント「RecommendIntent」に紐づくサンプル発話を2~3フレーズ考えて入力し、「モデルを保存」ボタンをクリックします。



サンプル発話を入力する際、疑問符(?)、英数文字、記号などは入力しないでください。

5. 続いて、左パネルのインテントのグループに「ビルトインインテント(3)」が含まれていることを確認します。AMAZON.HelpIntent をクリックします。

6. 同様に、AMAZON.CancelIntent、AMAZON.StopIntent があることも確認します。これらは、スキル作成時にデフォルトで追加されているビルトイン・インテントです。対話モデルでは、特に何もする必要はありませんが、必要に応じて独自のサンプル発話を追加することもできます。一方、これらのインテントを受け取るLambdaのコードは必ず実装する必要があります。



2.2.2. LambdaでRecommendIntent インテントandlerを追加する



プログラミングに自信のない方は、Lambdaの **index.js** のコードをサンプルファイル **[Sample2.js]** のコードに置き換えてください。

1. AWSコンソール画面を開き、CoffeeshopスキルのLambda関数コード **index.js** を開きます。

```

function handler(event, context) {
    const response = {
        version: "1.0",
        response: {
            speech: "Hello, how can I help you today?",
            reprompt: "I'm here to help you with your coffee orders.",
            cardTitle: "Welcome to CoffeeShop!",
            cardContent: "You can say 'Order coffee', 'Cancel order', or 'Get recommendations'.",
            icon: "https://www.coffeeshop.com/icon.png"
        }
    };
    const { request } = JSON.parse(event);
    if (request.type === "IntentRequest" && request.intent.name === "OrderIntent") {
        return OrderIntentHandler(request);
    } else if (request.type === "IntentRequest" && request.intent.name === "RecommendIntent") {
        return RecommendIntentHandler(request);
    } else if (request.type === "IntentRequest" && request.intent.name === "HelpIntent") {
        return HelpIntentHandler(request);
    } else if (request.type === "IntentRequest" && request.intent.name === "StopIntent") {
        return StopIntentHandler(request);
    } else {
        return response;
    }
}

const OrderIntentHandler = {
    canHandle(handlerInput) {
        return handlerInput.requestEnvelope.request.type === 'IntentRequest'
            && handlerInput.requestEnvelope.request.intent.name === 'OrderIntent';
    },
    handle(handlerInput) {
        const speechOutput = 'コーヒーですね、ありがとうございます。今日は天気がいいので全部100円です';
        return handlerInput.responseBuilder
            .speak(speechOutput)
            .getResponse();
    }
};

// この下に RecomendIntentHandler のコードを追加してください。
// この下に HelpIntentHandler のコードを追加してください。

```

2. OrderIntentHandlerが定義されているブロックをコピーし、「// この下に RecomendIntentHandlerのコードを追加してください。」と書かれているコメント行の下に貼り付け、以下のようにハンドラーネーム(RecommendIntent)、canHandle内のインテント名(RecommendIntent)、speechOutputに代入されている文字列を修正します。

リスト 3. OrderIntentをコピーして追加修正したRecommendIntentのコード

```
const RecommendIntentHandler = {
  canHandle(handlerInput) {
    return handlerInput.requestEnvelope.request.type === 'IntentRequest'
      && handlerInput.requestEnvelope.request.intent.name === 'RecommendIntent';
  },
  handle(handlerInput) {
    const speechOutput = '今日は甘さスッキリのカフェラテがおすすめです。';
    return handlerInput.responseBuilder
      .speak(speechOutput)
      .getResponse();
  }
};
```

- インテントandlerを追加したら、コードの下方に定義されている **SkillBuilder** オブジェクトに、先ほど追加した**RecommendIntentHandler**を登録します。

```
const skillBuilder = Alexa.SkillBuilders.custom();
exports.handler = skillBuilder
  .addRequestHandlers(
    LaunchRequestHandler,
    OrderIntentHandler,
    // ここにRecommendIntentHandlerを追加登録してください。
    RecommendIntentHandler, //<= 追加
    // ここにHelpIntentHandlerを追加登録してください。
    CancelAndStopIntentHandler,
    SessionEndedRequestHandler
  )
  .addErrorHandlers(ErrorHandler)
  .lambda();
```

- 同様の手順でビルトインインテント**AMAZON.HelpIntent**を処理する**HelpIntentHandler**を定義します。以下のコードを **RecommendIntent** の下に追加します。

```
const HelpIntentHandler = {
  canHandle(handlerInput) {
    return handlerInput.requestEnvelope.request.type === 'IntentRequest'
      && handlerInput.requestEnvelope.request.intent.name === 'AMAZON.HelpIntent';
  },
  handle(handlerInput) {
    const speechOutput =
      'コーヒーショップです。挽きたての美味しいコーヒーをお届けしています。今日は何にしますか?';
    const reprompt = '今日は何にしますか?';
    return handlerInput.responseBuilder
      .speak(speechOutput)
      .reprompt(reprompt)
      .getResponse();
  }
};
```

OrderIntentHandler, RecommendIntentHandlerと異なり、レスポンスを返す処理の記述に、**.reprompt(文字列)** が追加されていることに注目してください。AMAZON.HelpIntentは、ユーザーが使い方を尋ねてきた場合のインテントです。スキルは、スキルの使い方を簡潔に伝えた後、次にどうするかをユーザーに尋ね返事待ちの状態にする必要があります。**.reprompt(文字列)** の行を追加すると、はじめに**.speak(文字列)** の文字列を話した後、Alexaはユーザーからの応答待ちになります（ブルーのリングライトが回転します）。8秒間ユーザーの応答がなかった場合、Alexaは**.reprompt(文字列)** の文字列の内容を話し、さらにユーザーからの応答を待ちます。それでもユーザーからの応答がなかった場合、スキルは終了します。



5. インテントハンドラーを追加したので、コードの下方に定義されている SkillBuilderオブジェクトに、HelpIntentHandlerを登録します。

```

const skillBuilder = Alexa.SkillBuilders.custom();
exports.handler = skillBuilder
  .addRequestHandlers(
    LaunchRequestHandler,
    OrderIntentHandler,
    // ここにRecommendIntentHandlerを追加登録してください。
    RecommendIntentHandler,
    // ここにHelpIntentHandlerを追加登録してください。
    HelpIntentHandler, // <== 追加
    CancelAndStopIntentHandler,
    SessionEndedRequestHandler
  )
  .addErrorHandlers(ErrorHandler)
  .lambda();

```

i **skillBuiler**オブジェクトにハンドラーを登録する際の順番に注意してください。SDKは、登録された順にハンドラーの **canHandle()** をチェックし、最初に True を返したハンドラーの **handle()**を実行します。

2.3. テストする

対話モデル及びLambda関数の修正が終わったらスロットの値が正しく取得できるかどうかテストしましょう。

1. Alexaシミュレータを開き、「**コーヒーショップを開いて、おすすめは何**」のように発話を入力してテストしてみましょう。
2. 次に「**コーヒーショップを開いて**」だけでスキルを起動してみましょう。インテントタイプ **LaunchRequest** が送信され、**LaunchRequestHandler** が実行され、ユーザーからの返答待ちになることを確認しましょう。
3. そのままの状態で、Alexaに「**ヘルプ**」と言ってみましょう。**AMAZON.HelpIntent** が受信され、**HelpIntentHandler** が実行されることを確認してください。

以上のような手順で、インテントを追加するごとにインテントハンドラーを定義して**skillBuilder**オブジェクトに追加登録するのが ASK SDK for Node.js の基本パターンとなります。

ただし、サンプルコードの**CancelAndStopIntentHandler**のように、インテントが異なっていても、処理の内容や応答が同じ場合には、1つのインテントハンドラーに処理をまとめることもできます。



必ずしも、インテント:インテントハンドラーは1:1である必要はない。

リスト 4. CancelAndStopIntentHandler インテントハンドラーの定義例

(AMAZON.CancelIntentとAMAZON.StopIntentが同じハンドラーで処理されている)

```
const CancelAndStopIntentHandler = {
  canHandle(handlerInput) {
    return handlerInput.requestEnvelope.request.type === 'IntentRequest'
      && (handlerInput.requestEnvelope.request.intent.name === 'AMAZON.CancelIntent'
        || handlerInput.requestEnvelope.request.intent.name === 'AMAZON.StopIntent');
  },
  handle(handlerInput) {
    const speechOutput = 'コーヒーショップをご利用ありがとうございました。';
    return handlerInput.responseBuilder
      .speak(speechOutput)
      .getResponse();
  }
};
```

3. スロットの使用

3.1. この実習の学習目標

この実習ではユーザーの発話から ドリンクのメニューと ドリンクの数の情報を受け取るようにスキルを改良します。

この実習で学んだテクニックを使うと次のようなことができるようになります。

- 対話モデルのサンプル発話にスロットを導入する手順を実演できる
- 標準スロットタイプとカスタムスロットタイプを使い分けることができる。
- スロットで受け取ったデータがどのようにエンドポイントに渡されるかを説明できる。

3.2. 対話モデルの編集

例 3. スキルの会話サンプル



「アレクサ、コーヒーショップを開いて、カフェラテを二つください」



「カフェラテを2つですね、ありがとうございます。今日は天気がいいので全部100円でいいですよ。またのご利用をお待ちしております。」



- サンプル発話に「スロット」を追加し、エンドポイントでインテンントと共にスロットの値を受け取れるようにします。

参考ドキュメント



- インtent、発話、スロットの作成
- スロットタイプリファレンス
- カスタムスロットタイプの作成と編集



第5回 Alexa道場：スロットを使って発話内の可変文字列を取得しよう

3.2.1. スロットを追加する

1. 開発者コンソールに戻りスキルビルダーを開きます。
2. インテント「**OrderIntent**」をクリックします。
3. スロットを含むサンプル発話を追加します。まず「コーヒーを一つ注文して」と入力します。



インテント / OrderIntent

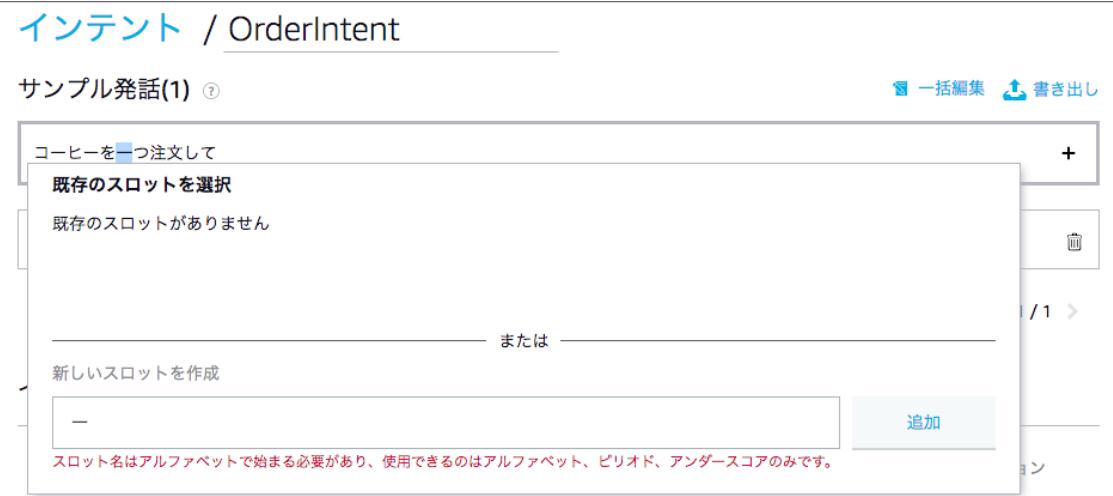
サンプル発話(1) ①

コーヒーを一つ注文して

コーヒーを注文して

1-1 / 1

4. スロットにしたい部分「一」（漢数字の1）マウスで選択します。下図のようなスロットを追加するポップアップが表示されます。



インテント / OrderIntent

サンプル発話(1) ①

コーヒーを一つ注文して

既存のスロットを選択

既存のスロットがありません

または

新しいスロットを作成

一

スロット名はアルファベットで始まる必要があります、使用できるのはアルファベット、ピリオド、アンダースコアのみです。

追加

1 / 1

5. 新しいスロットを作成のテキストフィールドに、追加したいスロット名「amount」を入力しましょう。スロット名は必ず半角英文字で入力します。入力したら「追加」ボタンをクリックします。

インテント / OrderIntent

サンプル発話(1) ①

一括編集 書き出し

コーヒーを一つ注文して

既存のスロットを選択

既存のスロットがありません

———— または ————

新しいスロットを作成

amount 追加

スロットタイプ ② アクション

6. スロットを含むサンプル発話に置き換えられています。+ボタンか、リターンキーを押して確定します。

インテント / OrderIntent

サンプル発話(2) ①

一括編集 書き出し

このインテントの呼び出しに使われると思われる発話

コーヒーを {amount} つ注文して

コーヒーを注文して

7. 画面の下の方を見ると、インテントスロットにamountが追加されていることが確認できます。amountのスロットタイプを設定しましょう。この場合「数値」を取得するので、標準スロットタイプのAMAZON.NUMBERを選択します。

インテントスロット(1) ①

順序 ②

名前 ②

スロットタイプ ②

アクション

1

amount

スロットタイプを選択

ダイアログを編集 削除

2

新しいスロットを作成

AMAZON.Month

ダイアログを編集 削除

AMAZON.Movie

AMAZON.NUMBER

AMAZON.Person

AMAZON.Reason

インテントの確認

このインテントには確認が必要ですか? ③

==== カスタムスロットタイプを追加する

8. 左のパネルからスロットタイプの「追加」ボタンをクリックします。

日本語

モデルを保存 モデルをビルト

カスタム 対話モデル

呼び出し名 インテント(5) + 追加

OrderIntent

amount

サンプル発話(2)

このインテントの呼び出しに使われると思われる発話

コーヒーを {amount} つ注文して

コーヒーを注文して

一括編集 書き出し

1 - 2 / 2

インテントスロット(1)

順序	名前	スロットタイプ	アクション
1	amount	AMAZON.NUMBER	ダイアログを編集 削除

9. カスタムスロットタイプ名を **MenuList** にして、「カスタムスロットタイプを作成」ボタンをクリックします。

スロットタイプを追加

スロットタイプは、インテントスロット内のデータの認識方法および処理方法を定義します。すべてのスロットをスロットタイプに割り当てる必要があります。スロットのタイプの使用についての詳細は[こちら](#)。

カスタムスロットタイプを作成

MenuList カスタムスロットタイプを作成

アレクサのビルトインライブラリから既存のスロットタイプを使用

ビルトインスロットタイプの使用についての詳細は[こちら](#)。

ビルトインを検索

10. スロット値を入力する画面が開きます。コーヒーショップで選択できるメニュー品目のリストを追加します。ここでは「コーヒー」「カフェオレ」「カプチーノ」「エスプレッソ」の4つを追加しましょう。追加が終わったら「モデルを保存」ボタンをクリックしておきます。

モデルを保存 モデルをビルト

スロットのタイプ / MenuList

スロット値(4) 検索

このスロットタイプの新しい値を入力 +

値	ID (オプション)	同義語(オプション)
エスプレッソ	IDを入力	同義語を追加 +
カプチーノ	IDを入力	同義語を追加 +
カフェラテ	IDを入力	同義語を追加 +
コーヒー	IDを入力	同義語を追加 +

1 - 4 / 4 < >

11. インテント OrderIntentの編集画面を表示し、既に登録されているサンプル発話の「コーヒー」と書かれている部分をマウスで選択します。

インテント / OrderIntent

サンプル発話(2) 一括編集 書き出し

このインテントの呼び出しに使われると考えられる発話 +

コーヒーを {amount} つ注文して

既存のスロットを選択

- amount

または

新しいスロットを作成

コーヒー 追加

スロット名はアルファベットで始まる必要があり、使用できるのはアルファベット、ピリオド、アンダースコアのみです。

12. 新しいスロットを作成のテキストフィールドに、追加するスロット名menuを入力し、「追加」ボタンをクリックします。

インテント / OrderIntent

サンプル発話(2) ?

一括編集 書き出し

このインテントの呼び出しに使われると思われる発話

コーヒーを {amount} つ注文して

既存のスロットを選択

amount

または

新しいスロットを作成

menu 追加

スロットタイプ ? アクション

◀ 1 - 2 / 2 ▶

13. 下方のインテントスロットを確認します。2行目を1クリックすると新しいスロットmenuが表示されます。スロットmenuのスロットタイプ **MenuList** をスロットタイプのリストから選択します。

インテントスロット(2) ?

順序 ?	名前 ?	スロットタイプ ?	アクション
1	amount	AMAZON.NUMBER	ダイアログを編集 削除
2	menu	スロットタイプを選択 AMAZON.NUMBER MenuList	ダイアログを編集 削除
3	新しいスロットを作成	+ AMAZON.Actor AMAZON.Animal AMAZON.Artist	ダイアログを編集 削除

インテントの確認

このインテントには確認が必要ですか? ?

14. 他の既存のサンプル発話の「コーヒー」の部分もスロットmenuに置き換えます。

インテント / OrderIntent

サンプル発話(2) ②

このインテントの呼び出しに使われると考えられる発話

{menu} を {amount} つ注文して

コーヒーを注文して

既存のスロットを選択

amount
menu

または

新しいスロットを作成

コーヒー

追加

スロット名はアルファベットで始まる必要があります。使用できるのはアルファベット、ピリオド、アンダースコアのみです。

15. 全ての「コーヒー」をスロットmenuに置き換えた後、「モデルを保存」ボタンと「モデルをビルト」ボタンを順にクリックしましょう。

alexa developer console

日本語 モデルを保存 モデルをビルト

カスタム 対話モデル

呼び出し名 インテント(5) OrderIntent ピルトインテント(4) AMAZON.CancelIntent AMAZON.HelpIntent AMAZON.StopIntent AMAZON.NavigateHomeIntent スロットタイプ(2) AMAZON.NUMBER

インテント / OrderIntent

サンプル発話(2) ②

このインテントの呼び出しに使われると考えられる発話

{menu} を {amount} つ注文して

{menu} を注文して

インテントスロット(2) ②

順序 ①	名前 ①	スロットタイプ ①	アクション
1	amount	AMAZON.NUMBER	ダイアログを編集 削除

以上で、対話モデルの修正は完了です。次にLambda関数の修正に移りましょう。

3.3. Lambdaのコードを改良する

スロットmenuの値を受け取れるようにLambda側のプログラムコードも修正しましょう。



プログラミングに自信のない方は、Lambdaの `index.js` のコードをサンプルファイル [\[Sample3.js\]](#) のコードに置き換えてください。

リスト 5. スロット **menu** と **amount** の値を取得し、それぞれの値を含めた応答を返すコード

```
var amount = handlerInput.requestEnvelope.request.intent.slots.amount.value;
var menu = handlerInput.requestEnvelope.request.intent.slots.menu.value;

const speechOutput = menu + 'を' + amount + 'つですね、ありがとうございます。今日は天気がいいので全部100円でいいですよ。またの御利用をお待ちしております。';

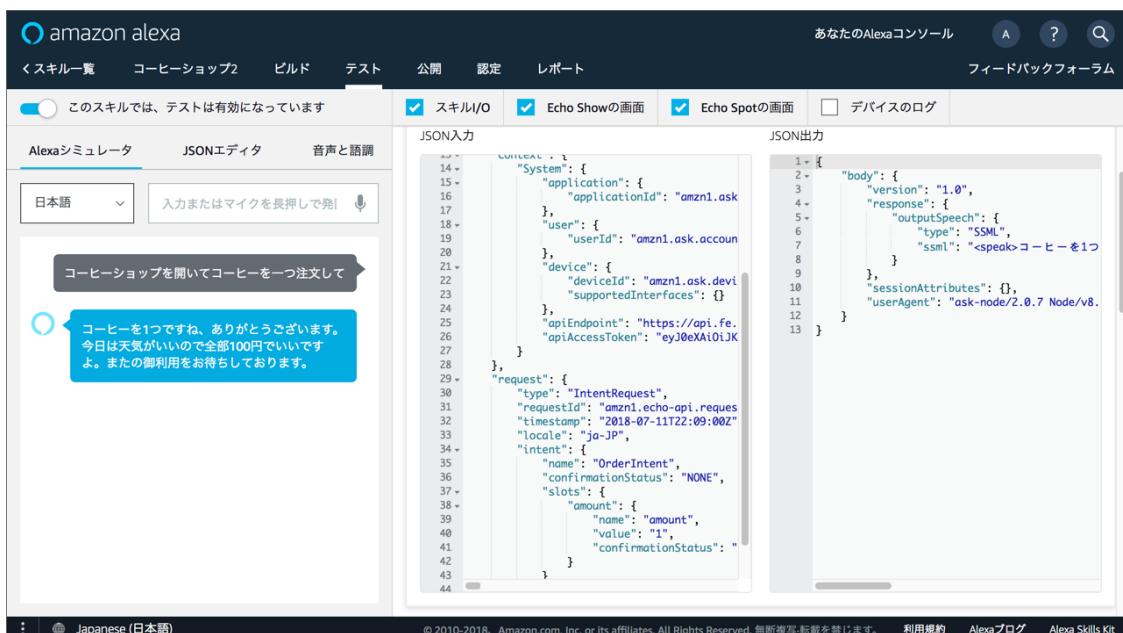
return handlerInput.responseBuilder
.speak(speechOutput)
.getResponse();
```

Lambda関数のコードの修正が完了したら「保存」ボタンをクリックしてテストしましょう。

3.4. テストする

対話モデル及びLambda関数の修正が終わったらスロットの値が正しく取得できるかどうかテストしましょう。

- Alexaシミュレータを開き、「コーヒーショップを開いて、コーヒーを一つ注文して」のようにスロットを含めた発話を入力してテストしてみましょう。このとき、スロットに数字を入れる場合は、漢数字で入れてください。半角や全角の数字はうまく認識しないので注意してください。タイプ入力だけではなく、音声入力でもテストを繰り返し、スロットの値が正しく取得できているかを確認しましょう。



- JSON入力で、スロットの値が正しく取得できているかどうかを観察します。下図のようにスロット名 **menu** にユーザーが発話したメニューの名前（下図の場合は コーヒー）、**amount** に数（下図の場合は数字の 2）が入ることを確認します。

スキルI/O

JSON入力

```
29 "request": {  
30     "type": "IntentRequest",  
31     "requestId": "amzn1.echo-api.request.abf112d9-4c8a-4f22-be48-d26cd9463254",  
32     "timestamp": "2018-10-25T09:46:48Z",  
33     "locale": "ja-JP",  
34     "intent": {  
35         "name": "OrderIntent",  
36         "confirmationStatus": "NONE",  
37         "slots": {  
38             "menu": {  
39                 "name": "menu",  
40                 "value": "コーヒー",  
41             },  
42             "resolutions": [  
43                 "resolutionsPerAuthority": [  
44                     {  
45                         "authority": "amzn1.er-authority.echo-sdk.amzn1.ask.skill",  
46                         "status": {  
47                             "code": "ER_SUCCESS_MATCH"  
48                         },  
49                         "values": [  
50                             {  
51                                 "value": {  
52                                     "name": "コーヒー",  
53                                     "id": "Zeebd23884f737c016a7e1145451d5fb"  
54                                 }  
55                             }  
56                         ]  
57                     }  
58                 ],  
59             },  
60             "confirmationStatus": "NONE",  
61             "source": "USER"  
62         },  
63         "amount": {  
64             "name": "amount",  
65             "value": "2",  
66             "confirmationStatus": "NONE",  
67             "source": "USER"  
68         }  
69     },  
70 },  
71 }
```

3. JSON出力側も正しく数字の入った応答を返しているかを確認しましょう。

4. 発話のバリエーション

4.1. この実習の学習目標

この実習では、ユーザーの様々な発話パターンをAlexaが正しく認識できるように改良するテクニックを学びます。

この実習で学んだテクニックを使うと次のようなことができるようになります。

- 対話モデルのインテントに適切な数のサンプル発話を追加して、ユーザーの発話の認識率を向上させることができる。
- スロットに同義語（シノニム）を追加して、同じ意味をもつ別の言い方にも対応できるように改良できる。

4.2. 対話モデルの編集

例 4. スキルの会話サンプル



「アレクサ、コーヒーショップで、冷たいコーヒーを一つ頼む」



「冷たいコーヒーを1つですね。ありがとうございます。今日は天気がいいので全部100円でいいですよ。またの御利用をお待ちしております。」



「アレクサ、コーヒーショップを開いて、カフェラテをください」



「カフェラテですね。ありがとうございます。今日は天気がいいので全部100円でいいですよ。またの御利用をお待ちしております。」

参考ドキュメント



- インテント、発話、スロットの作成
- スロットタイプ値の同義語とIDを定義する（エンティティ解決）

4.2.1. インテントのサンプル発話を追加する

インテントに対するユーザーの発話パターンは一つとは限りません。同じ内容のことを話す場合でも、人によって様々な言い回しをします。住む地域によっては方言もあるでしょう。これら発話の多様性に対応できるようスキルを改良しましょう。

1. 開発者コンソールのスキルビルダーを開きます。
2. インテント「**OrderIntent**」をクリックして、サンプル発話を追加します。言葉の流れ、異なる言い回し、方言なども考慮すると良いでしょう。また、全てのスロットが含まれる場合、一部のスロットしか含まれない場合、スロットが全く含まれない場合、スロットの順番が異なる場合など、あらゆる発話パターンを想定し、できるだけ多く入力します。



サンプル発話に入力したフレーズは、Alexaの機械学習の教師データとして登録されます。サンプル発話の数が多ければ多いほど、認識率が向上します。

サンプル発話
{menu}
{menu} を {amount} フブリーズ
{menu} を {amount} 杯ブリーズ
{menu} を {amount} 杯注文して
{menu} を {amount} つ注文して
{menu} を {amount} 杯頼む
{menu} を {amount} 杯ください
{menu} を {amount} つ頼む
{menu} を {amount} つちょうだい
{menu} を {amount} つください
{menu} ください
{menu} を注文して
{menu} が欲しい
{menu} にしようかな

3. 同様に、インテント「**RecommendIntent**」をクリックして、サンプル発話を追加します。
4. 対話モデルを修正したら対話モデルを保存し、ビルドします。
5. AlexaシミュレーターまたはAlexa対応デバイスでテストをします。

4.2.2. 同義語（シノニム）を追加する

同じ意味を持つ言葉でも様々なフレーズを使います。例えば「アイスコーヒー」のことを「冷たいコーヒー」という人もいるでしょう。「カフェオレ」という人もいれば「カフェオーレ」という人もいるでしょう。こうした言葉の揺れはスロット値の取得に影響を及ぼします。異なるフレーズでも同じスロット値として取り扱いたい場合には、シノニムの機能を使うと便利です。

ここでは、既に登録済みの **MenuList** スロットタイプの各値にシノニムを追加しましょう。

1. 開発者コンソールのスキルビルダーを開きます。
2. 左パネルの「スロットタイプ」から **MenuList** をクリックします。
3. 既に登録されているドリンクメニューの名前に対し、それぞれ別の言い回しを追加します。
4. プログラムコードで処理しやすいように各スロット値に固有のIdを付与することもできます。Idは任意で入力します。
5. 対話モデルを修正したら対話モデルを保存し、ビルドします。
6. AlexaシミュレーターまたはAlexa対応デバイスでテストをします。

4.3. Lambdaのコードを改良する



プログラミングに自信のない方は、Lambdaの **index.js** のコードをサンプルファイル **[Sample4.js]** のコードに置き換えてください。

4.3.1. スロット値の有無によって応答を変える

例えば **OrderIntent** に注目した場合、ユーザーは必ずしもメニューと個数を言ってくれるとは限りません。もし発話の中でスロット値が提供されなかった場合、リクエストデータにはそのスロット値は含まれず、Node.js のスロット変数の値は **undefined** となります。

コードの中ではこのような場合のエラー処理も考慮する必要があります。例えば下のようにコードを修正します。

リスト 6. スロット値の有無によって応答を変えるコードの例

```
const OrderIntentHandler = {
  canHandle(handlerInput) {
    return handlerInput.requestEnvelope.request.type === 'IntentRequest'
      && handlerInput.requestEnvelope.request.intent.name === 'OrderIntent';
  },
  handle(handlerInput) {
    var amount = handlerInput.requestEnvelope.request.intent.slots.amount.value;
    var menu = handlerInput.requestEnvelope.request.intent.slots.menu.value;
    if (menu === undefined){
      // ユーザーがメニューを言わなかった場合
      const speechOutput =
        'コーヒー、カフェラテ、カプチーノからお選びいただけます。どれにしますか？';
      const reprompt = '何にしますか？';
      return handlerInput.responseBuilder
        .speak(speechOutput)
        .reprompt(reprompt)
        .getResponse();
    } else if(amount === undefined){
      // ユーザーが数量を言わなかった場合
      const speechOutput = menu + 'ですね。ありがとうございます。今日は天気がいいので全部
      100円でいいですよ。またの御利用をお待ちしております。';
      return handlerInput.responseBuilder
        .speak(speechOutput)
        .getResponse();
    } else {
      // ユーザーがメニューと数量の両方を言った場合
      const speechOutput = menu + 'を' + amount +
        'つですね、ありがとうございます。今日は天気がいいので全部100円でいいですよ。またの御利用をお待ちしております。';
      return handlerInput.responseBuilder
        .speak(speechOutput)
        .getResponse();
    }
  }
};
```

4.3.2. 同義語に含まれるフレーズかどうかを確認する（エンティティ解決）

ユーザーが話したスロット値が、同義語に含まれるフレーズかそうではないかは、**slots** データに含まれるステータスコード **<slot名>.resolutions.resolutionsPerAuthority**
[].status.code の値を調べることで判別できます。

ステータスコードの意味は、次の通りです。

- ユーザーの値がカスタム値または同義語の1つに一致した場合、ステータスコードは**ER_SUCCESS_MATCH**になります。
- ユーザーの値がそれ以外の値（そのタイプのビルトインデータなど）に一致した場合、ステータスコードは**ER_SUCCESS_NO_MATCH**になります。

ステータスコードを調べ、同義語に含まれる場合、そのスロット値の代表値で応答を返すサンプルコードは以下のようになります。お好みでLambdaコードに組み込んでください。

リスト 7. 同義語に含まれる場合は、その代表値で応答を返すコードの例

```
// ステータスコードを取得する
let status = handlerInput.requestEnvelope.request.intent.slots.menu.resolutions.resolutionsPerAuthority[0].status.code;
if (status === "ER_SUCCESS_MATCH"){
    // ステータスコードを調べて、MATCHだったら代表値を取得してmenuにセットする
    menu = handlerInput.requestEnvelope.request.intent.slots.menu.resolutions.resolutionsPerAuthority[0].values[0].value.name;
}
```

4.4. テストする

対話モデル及びLambda関数の修正が終わったら、あらゆるユーザーの発話のバリエーションでも正しくスキルが動作するかどうかテストしましょう。

1. Alexaシミュレータを開き、様々な発話のバリエーションを試してください。テキストのタイプ入力だけでなく、音声入力でも試してください。



図 1. 様々な発話のバリエーションでテストする

2. カスタムスロットタイプのリストに含まれるメニューを言った場合と、含まれないメニューを言った場合のJSONデータの違いを確認してください。

The screenshot shows the Alexa Skills Kit (ASK) developer console. At the top, there is a blue speech bubble icon followed by the text "カフェラテを2つですね、ありがとうございます。今日は天気がいいので全部100円でいいですよ". Below this, there is a section titled "スキルI/O" with a "JSON入力" tab selected. The JSON code is displayed in a code editor-like interface:

```
35     "name": "OrderIntent",
36     "confirmationStatus": "NONE",
37     "slots": {
38       "menu": {
39         "name": "menu",
40         "value": "カフェオーレ",
41         "resolutions": {
42           "resolutionsPerAuthority": [
43             {
44               "authority": "amzn1.er-authority.echo-sdk.amzn1.c",
45               "status": {
46                 "code": "ER_SUCCESS_MATCH"
47               },
48               "values": [
49                 {
50                   "value": {
51                     "name": "カフェラテ",
52                     "id": "cafelatte"
53                   }
54                 }
55               ]
56             }
57           ],
58         },
59         "confirmationStatus": "NONE",
60         "source": "USER"
61       },
62       "amount": {
63         "name": "amount",
64         "value": "2",
65       }
66     }
```

図 2. カスタムスロットタイプの同義語に含まれる場合

The screenshot shows the Amazon Alexa developer console interface. At the top, there is a blue speech bubble icon followed by the text "抹茶ラテを1つですね、ありがとうございます。今日は天気がいいので全部100円でいいですよ". Below this, there is a section titled "スキルI/O" with a "JSON入力" tab selected. The JSON code is displayed in a code editor window:

```
34  "intent": {  
35      "name": "OrderIntent",  
36      "confirmationStatus": "NONE",  
37      "slots": {  
38          "menu": {  
39              "name": "menu",  
40              "value": "抹茶 ラテ",  
41              "resolutions": {  
42                  "resolutionsPerAuthority": [  
43                      {  
44                          "authority": "amzn1.er-authority.echo-sdk.amzn1.c",  
45                          "status": {  
46                              "code": "ER_SUCCESS_NO_MATCH"  
47                          }  
48                      }  
49                  ]  
50              },  
51              "confirmationStatus": "NONE",  
52              "source": "USER"  
53          },  
54          "amount": {  
55              "name": "amount",  
56              "value": "1",  
57              "confirmationStatus": "NONE",  
58              "source": "USER"  
59          }  
60      }  
61  }  
62 }  
63 }
```

図3. カスタムスロットタイプの同義語に含まれない場合

5. セッションアトリビュート

メニューの品目と数量を取得できたら、お砂糖が必要かどうかをユーザーに尋ねるスキルを作ってみましょう。

課題3までのコーヒーショップスキルは、ユーザーが注文するフレーズを話し、Alexaが応答してスキルが終了するという、一回のインタラクションしかないシンプルなものでした。

これに、Alexaが「砂糖をおつけしますか？」という追加の質問を加えることで、複数回の対話を制御するコードが必要になります。

5.1. この実習の学習目標

コーヒーショップスキルの対話モデルに「砂糖をつけるかどうか」の質問を追加してマルチターンのスキルに改造します。

この実習で学習したテクニックを使うと、以下のことができるようになります。

- セッションアトリビュートを使って過去の対話で取得した情報を保持しておくテクニックを使えるようになる。
- ユーザーのYES/NOの意思を取得するビルトインインテント **AMAZON.YesIntent**、**AMAZON.NoIntent** を使えるようになる。

参考ドキュメント

- カスタムスロットタイプの作成と編集

5.2. 会話のサンプル

例 5. サンプル1



「アレクサ、コーヒーショップを開いて、コーヒーを2つください」



「コーヒーを2つですね、ありがとうございます。お砂糖をおつけしますか？」



「はい」



「コーヒーを2つ、お砂糖をつけて ご用意いたします。ご利用ありがとうございます。」

さらに、もしお客様が数量を言ってくれなかつた場合にも、それを尋ねる処理も加えましょ

う。

例 6. サンプル2

アレクサ、「アレクサ、コーヒーショップを開いて、コーヒーをください」

アレクサ、「コーヒーですね。おいくつご用意しましょうか？」

アレクサ、「二つお願い」

アレクサ、「コーヒーを2つですね。お砂糖はおつけしますか？」

アレクサ、「はい」

アレクサ、「コーヒーを2つ、お砂糖をつけてご用意いたします。ご利用ありがとうございます。」

かなり自然な会話に近づきますね？これをどのようにスキルで実現するかを学習しましょう。

5.3. Lambdaのコードを改良する



プログラミングに自信のない方は、Lambdaの **index.js** のコードをサンプルファイル [\[Sample5.js\]](#) のコードに置き換えてください。

自力でプログラミングしたい方は、以下のヒントを元にコードを修正してください。

- 「コーヒーを2つですね。ありがとうございます」と返して終了する部分を、「コーヒーを2つですね。お砂糖はおつけしますか？」とユーザーに尋ねるように変更しましょう。

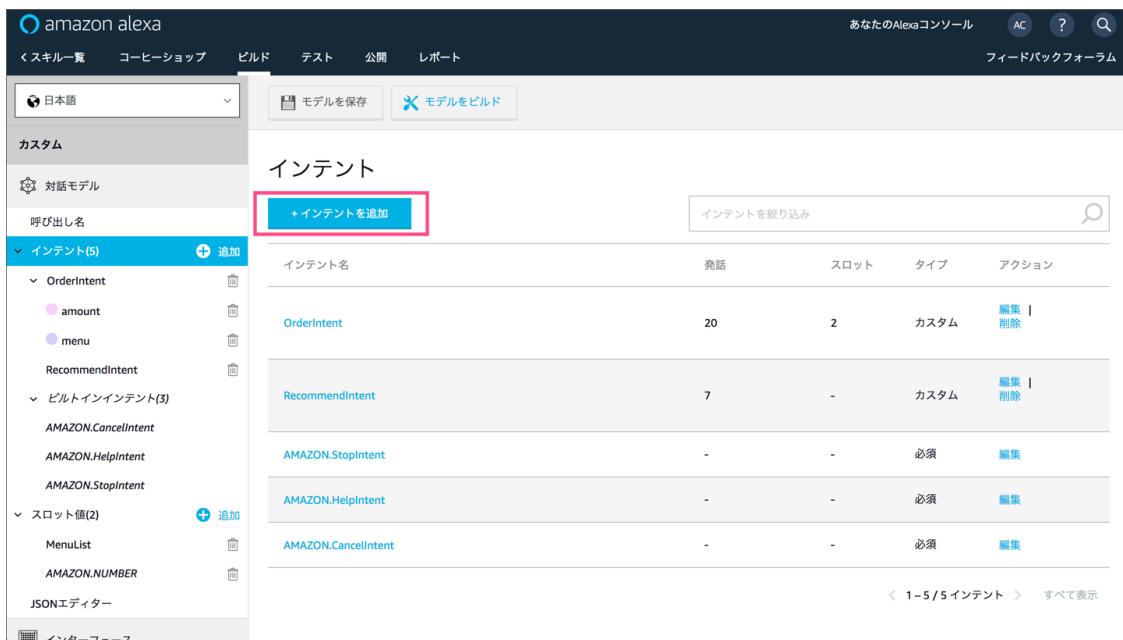
リスト 8. 変更前のコード

```
const speechOutput = menu + 'を' + amount +  
  'つですね、ありがとうございます。今日は天気がいいので全部100円でいいですよ。またの御利用をお待ち  
  しております。';  
return handlerInput.responseBuilder  
  .speak(speechOutput)  
  .getResponse();
```

リスト 9. 変更後のコード

```
const speechOutput = menu + 'を' + amount + 'つですね。お砂糖はおつけしますか？';
const reprompt = 'お砂糖はおつけしますか？';
return handlerInput.responseBuilder
.speak(speechOutput)
.reprompt(reprompt)
.getResponse();
```

2. 「お砂糖をおつけしますか？」を聞かれた後、ユーザーは「はい」か「いいえ」と言った応答をしてくるでしょう。これを受け取るインテントも用意しておきます。これには標準インテントの **AMAZON.YesIntent**、**AMAZON.NoIntent** を利用します。スキルビルダーに追加しておきましょう。
3. スキルビルダーのインテント画面を開き「インテントを追加」ボタンをクリックします。



4. 「アレクサのビルトインライブラリから既存のインテントを使用」を選択し、検索フィールドに **YES** とタイプします。すると、**AMAZON.YesIntent** が表示されます。「インテントを追加」ボタンをクリックします。

あなたのAlexaコンソール AC ? フィードバックフォーラム

日本語 モデルを保存 モデルをビルド

カスタム

対話モデル

呼び出し名

インテント(5) + 追加

- OrderIntent
 - amount
 - menu
- RecommendIntent
- ビルトインインテント(3)
 - AMAZON.CancelIntent
 - AMAZON.HelpIntent
 - AMAZON.StopIntent
- スロット値(2) + 追加
 - MenuList
 - AMAZON.NUMBER
- JSONエディター
- インターフェース

インテントを追加

インテントとは、ユーザーが話しかけたリクエストを実行するアクションのことです。インテントについての詳細は[こちら](#)。

○ カスタムインテントを作成 ①

インテントの名前を入力 カスタムインテントを作成

○ アレクサのビルトインライブラリから既存のインテントを使用 ②

ビルトインインテントの使用についての詳細は[こちら](#)。

ビルトイン名	説明
AMAZON.YesIntent	+インテントを追加

1/17個のビルトイン

- 左側のビルトインインテントのリストに **AMAZON.YesIntent** が追加されたことを確認します。必要に応じ **AMAZON.YesIntent** のサンプル発話を追加し、モデルを保存します。

あなたのAlexaコンソール AC ? フィードバックフォーラム

日本語 モデルを保存 モデルをビルド

カスタム

対話モデル

呼び出し名

インテント(6) + 追加

- OrderIntent
 - amount
 - menu
- RecommendIntent
- ビルトインインテント(4)
 - AMAZON.CancelIntent
 - AMAZON.HelpIntent
 - AMAZON.StopIntent
 - AMAZON.YesIntent
- スロット値(2) + 追加
 - MenuList
 - AMAZON.NUMBER
- JSONエディター

インテント / AMAZON.YesIntent

サンプル発話(4) ①

このインテントの呼び出しに使われると考えられる発話 +

よろしく
ください
頼むわ
お願い

< 1 - 4 / 4 > すべて表示

- 同様に、**AMAZON.NoIntent** も追加します。

The screenshot shows the 'Intent' section of the ASK console. On the left, there's a tree view of intents: '対話モデル' (Conversation Model) with 'AMAZON.NoIntent' selected, and other categories like 'OrderIntent', 'RecommendIntent', 'CancelIntent', 'HelpIntent', 'StopIntent', and 'YesIntent'. On the right, under 'サンプル発話(8)', there are five sample utterances. At the top right, there are buttons for 'モデルを保存' (Save Model) and 'モデルをビルド' (Build Model).

- はじめの会話で、メニュー品目と数量を取得し、Alexaは砂糖が必要かどうかを尋ねます。ユーザーがお砂糖の要不の回答を受信するインテント、**AMAZON.YesIntent** や **AMAZON.NoIntent** には、メニュー品目や数量を取得するスロットはありません。これらの情報はどこから入手するのでしょうか？

これには、セッションアトリビュートの機能を利用すると良いでしょう。セッションアトリビュートとは、ユーザーとの対話が継続している間（セッションと呼びます）、一時的に情報を保持するための機構です。情報を取得したタイミングで、以下のコードを追加し、セッションアトリビュートに保存しておきましょう。

リスト 10. セッションアトリビュートのデータを保存するコード

```
attributes.menu = menu;
attributes.amount = amount;
handlerInput.attributesManager.setSessionAttributes(attributes);
```

対話の中でこれらの情報が必要ななった時は、以下のようなコードでセッションアトリビュートから情報を取得しましょう。

リスト 11. セッションアトリビュートのデータを取り出すコード

```
const attributes = handlerInput.attributesManager.getSessionAttributes();
const amount = attributes.amount;
const menu = attributes.menu;
```

以上の修正ができたら対話モデルの変更を保存しビルドしましょう。Lambda関数も正しく保存されていることも確認しましょう。

5.4. テストする

でき上がったスキルをテストしましょう。Alexaシミュレータを開いて、マルチターンの会話がうまく動作するかテストしてください。

The screenshot shows the Alexa Skills Kit Test interface. At the top, there are tabs for 'スキル一覧' (Skills List), 'コーヒーショップ4' (CoffeeShop4), 'ビルト' (Built), 'テスト' (Test), '公開' (Public), '認定' (Certified), and 'レポート' (Report). The 'テスト' tab is selected. Below the tabs, there are checkboxes for 'スキル/I/O' (checked), 'Echo Showの画面' (checked), 'Echo Spotの画面' (checked), and 'デバイスのログ' (unchecked). The main area is divided into two sections: 'Alexaシミュレータ' (Alexa Simulator) on the left and '音声と語調' (Speech and TTS) on the right.

Alexaシミュレータ:

- Input: コーヒーショップを開いてカフェラテを一つ注文して
- Output: カフェラテを1つですね。お砂糖はおつけしますか？
- User Response: はい
- Output: カフェラテを1つ。お砂糖をつけてご用意いたします。ご利用ありがとうございました。

音声と語調:

日本語

スキルI/O:

JSON入力:

```

1: {
  "version": "1.0",
  "session": {
    "new": false,
    "sessionId": "amzn1.echo-api.session",
    "application": {
      "applicationId": "amzn1.ask.skill"
    },
    "attributes": {
      "amount": "1",
      "menu": "カフェラテ"
    },
    "user": {
      "userId": "amzn1.ask.account.A"
    }
  },
  "context": {
    "System": {
      "application": {
        "applicationId": "amzn1.ask.skill"
      },
      "user": {
        "userId": "amzn1.ask.account.A"
      }
    }
  }
}

```

JSON出力:

```

1: {
  "body": {
    "version": "1.0",
    "response": {
      "outputSpeech": {
        "type": "SSML",
        "ssml": "<speak>カフェラテを1つですね。お砂糖はおつけしますか？</speak>"
      }
    },
    "sessionAttributes": {
      "amount": "1",
      "menu": "カフェラテ"
    }
  },
  "userAgent": "ask-node/2.0.7 Node/v8"
}

```

補足

Alexaの開発者アカウントに関する問題

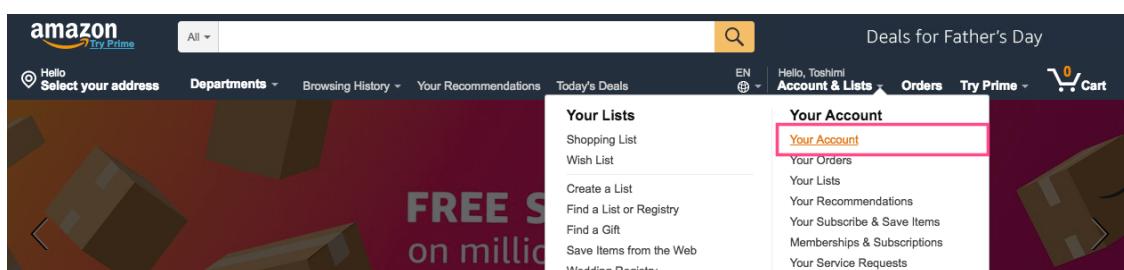
Alexaのスキル開発で使用する開発者アカウントは、Amazon Echoをセットアップした時に使用したアカウントと同じものを使用することを推奨します。作成したスキルを、同じアカウントでセットアップされたお手持ちのEchoデバイスすぐにテストできた方が便利だからです。

この時使用するAmazon.co.jpと同じアカウントがAmazon.comにも存在し、かつ同じパスワードが設定されている場合、作成したスキルが日本の Alexa.amazon.co.jp のスキル一覧画面に表示されなくなります。これは、作成したスキルが米国向けのスキルを日本語で作成した状況になり、alexa.amazon.com 上に作成されてあしまったからです。この状態では日本用にセットアップされた Echoデバイス上ではテストできないことになります。

対策1 Amazon.com のアカウントのパスワードを変更する

一つ目の対策は、Amazon.com のアカウントと Amazon.co.jp のアカウントのどちらかのパスワードを変更します。こうすることで開発者アカウントがどちらの国に所属するアカウントなのかを明確に切り分けることができるようになります。ここでは、Amazon.com の方のパスワードを変更する手順を示します。

1. <https://amazon.com> にアクセスして、ログインし、「Account & Lists」のメニューから「Your Account」をクリックします。



2. 「Login & Security」の枠内をクリックします。

The screenshot shows the Amazon account dashboard. At the top, there's a search bar and a deals banner for Father's Day. Below that, the user profile 'Toshimi's Amazon' is displayed along with links for 'AMAZON PRIME', 'AUDIBLE AUDIOBOOKS', 'Try Prime', and 'Try Audible'. The main area contains several boxes: 'Your Orders' (Track, return, or buy things again), 'Login & security' (Edit login, name, and mobile number - this box is highlighted with a red border), 'Your Addresses' (Edit addresses for orders and gifts), 'Payment options' (Edit or add payment methods), and 'Prime' (View benefits and payment settings). A 'Gift cards' box is also present.

3. 現在ログインしているアカウントのパスワードを入力します。

The screenshot shows the Amazon sign-in page. The title 'Sign in' is at the top. Below it is a 'Switch accounts' section with a user profile picture and the name 'Toshimi Hatanaka'. There are input fields for 'Password' and 'Forgot your password?'. A large yellow 'Sign in' button is centered below the password field. At the bottom, there's a checkbox for 'Keep me signed in.' followed by a 'Details' link.

4. Password:の項目の「Edit」ボタンをクリックします。

The screenshot shows the 'Login & security' settings page. It includes fields for 'Name' (Toshimi Hatanaka), 'Email' (redacted), 'Mobile Phone Number' (redacted), and 'Password' (*****). Each field has an 'Edit' button to its right. The 'Password' edit button is highlighted with a red box. Below these fields is a section for 'Advanced Security Settings' with a 'Manage how and when you receive security codes' link and an 'Edit' button.

5. パスワードを変更し「Save changes」ボタンをクリックします。

The screenshot shows the 'Change password' page. It has instructions: 'Use the form below to change the password for your Amazon account'. There are three input fields: 'Current password' (redacted), 'New password' (redacted), and 'Reenter new password' (redacted). Below these is a yellow 'Save changes' button.

6. 「Done」ボタンをクリックします。

The screenshot shows the 'Login & security' section of the Amazon Alexa account settings. A green success message box at the top says 'Success' and 'You have successfully modified your account!'. Below it, there are fields for Name (Toshimi Hatanaka), Email (redacted), Mobile Phone Number (redacted), and Password (*****). Each field has an 'Edit' button to its right. At the bottom of the section is an 'Advanced Security Settings' link with an 'Edit' button. A large orange 'Done' button is located at the bottom center of the page.

7. パスワードを変更したらWebブラウザを一度終了し再起動します。

8. Webブラウザが開いたら、<https://developer.amazon.com>にアクセスし、日本のアカウントとパスワードでログインします。

The screenshot shows the 'Amazon Developer' login page. It features a logo at the top left and a 'ログイン' (Login) button at the top center. Below the button are two input fields: 'Eメールまたは携帯電話番号' (Email or mobile phone number) and 'パスワード' (Password). To the right of the password field is a link 'パスワードを忘れた場合' (Forgot password?). Below these fields is a large orange 'ログイン' (Login) button. At the bottom of the form are two links: 'Amazon Developerの新しいお客様ですか?' (Are you a new customer?) and 'Amazon Developerアカウントを作成' (Create Amazon Developer account).

9. 開発者アカウントの新規登録画面が表示されたら、必要項目を入力し完了させます。

この時「国/リージョン」を必ず日本を選択してください。



ここで、開発者アカウントの登録画面が出てこなかった場合は、残念ながらパスワードでのアカウントの切り分けはうまく行かない可能性があります。その場合は対策2をお試しください。

申請

1. プロフィール情報 2. App Distribution Agreement 3. 支払い

*は必須項目を示しています。

国/リージョン * 日本

名 *

姓 *

Eメールアドレス *

電話番号 *

e.g. 212-555-1212, +44 0161 715 3369

Fax番号

開発者名 *

Amazon.co.jpのアプリに表示されます

開発者名 (ふりがな) *

開発者または会社名のふりがな

開発者概要

最大文字数: 4000, 残り: 4000

住所1 *

- 以上で、正しく日本のスキル用の開発者アカウントが作成されました。以降は、このアカウントを使ってスキルの開発を行います。

対策2 Amazon.com のアカウントのメールアドレスを別のものに変更する

既にUSのアカウントとして登録された開発者アカウントを別のメールアドレスに変更します。こうすることで、既存の amazon.co.jp のアカウントと開発者アカウントとの結び付きが解放され、新たに日本の開発者アカウントとして登録し直すことができます。以下に手順を示します。

- <https://developer.amazon.com> にアクセスし、USの開発者アカウントとして登録されてしまったアカウントでログインします。

amazon Developer

ログイン

Eメールまたは携帯電話番号

パスワード

パスワードを忘れた場合

ログイン

Amazon Developerの新しいお客様ですか?

Amazon Developerアカウントを作成

- ログインしたら「DeveloperPortal」のリンクをクリックします。

The screenshot shows the Amazon Developer Services and Technologies landing page. It features three main service sections: 'Alexa' (new technologies for natural, intuitive voice services), 'Amazonアプリストア' (Amazon Appstore for Fire TV, Fire Tablets, mobile platforms, and Android apps/games), and 'アマゾン ウェブ サービス' (Amazon Web Services for reliable, scalable, low-cost cloud computing). The Alexa section includes a sub-section for 'Beta test for Alexa skill is complete'.

3. 「設定」をクリックします。

The screenshot shows the Amazon Developer Console interface. The top navigation bar has tabs for 'ダッシュボード', 'アプリ & サービス', 'ALEXA SKILLS KIT', 'ALEXA VOICE SERVICE', 'レポート', 'サポート', '文書', and '設定'. The '設定' tab is highlighted with a red box. Below the navigation, there are two sections: '通知' (Notifications) and 'アナウンス' (Announcements). The '通知' section shows a single unread message about a beta test completion. The 'アナウンス' section lists an announcement from May 30, 2018, regarding new ways to improve Alexa skills.

4. Eメールアドレスの項目の右にある「編集」ボタンをクリックします。

The screenshot shows the 'My Account' settings page. Under the 'Eメールアドレス' (Email Address) section, there is a '権限' (Permissions) table. The '管理者' row has a checked checkbox. To the right of the table, a red box highlights the '編集' (Edit) button. Other rows in the table include 'マーケティング担当' (Marketing Manager), 'アナリスト' (Analyst), and '開発者' (Developer). At the bottom of the page, there are fields for 'パスワード' (Password) and 'Eメール配信' (Email Delivery), and a '保存' (Save) button.

5. ログインとセキュリティの画面でEメールアドレスの「編集」ボタンをクリックします。

amazon Developer

ログインとセキュリティ

名前
Toshimi Hatanaka 編集

Eメールアドレス 編集

携帯電話番号:
[携帯電話番号を追加する理由](#) 追加

現在のパスワード:
***** 編集

終了

6. Eメールアドレスを変更し「変更内容を保存」ボタンをクリックします。

amazon Developer

Eメールを変更する

Amazon DeveloperのアカウントのEメールアドレスを変更するには、下のフォームを使用してください。次回ログインまたは発注する際は、新しいEメールアドレスを使用してください。

現在のEメールアドレス：
[redacted]

新しいEメールアドレス：
[redacted] [copy]

新しいEメールアドレスの再入力：
[redacted]

現在のパスワード：
[redacted] [copy]

表示されている文字列を半角で入力してください
65xx2m

他の画像に切り替える

文字列を入力
[redacted]

視覚障害等により判読できない場合はこちら

変更内容を保存

7. 一度サインアウトし、再び開発者ポータルにログインします。この時、Eメールを変更する前の日本で使いたい開発者アカウントのEメールを入力します。

amazon Developer

ログイン

Eメールまたは携帯電話番号
[redacted]

パスワード [copy] パスワードを忘れた場合 [copy]

ログイン

Amazon Developerの新しいお客様ですか？ [redacted]

[redacted] Amazon Developerアカウントを作成

8. 開発者アカウントの新規登録画面が表示されたら、必要項目を入力し完了させます。
この時「国/リージョン」を必ず日本を選択してください。

9. 以上で、正しく日本のスキル用の開発者アカウントが作成されました。以降は、このアカウントを使ってスキルの開発を行います。

対策3 新規で amazon.co.jp のアカウントを作るところから始める

最も確実な方法は、開発用に amazon.co.jp のアカウントを新規で作成し、そのアカウントで開発者アカウントを登録します。この場合、既に既存のアカウントでセットアップされたEchoデバイスとは別アカウントになるので、Echoデバイスでテストしたい場合は、開発用のアカウントでEchoデバイスをセットアップし直す必要があります。

対策4 Amazon のサポートに連絡する

どうしてもアカウントの問題が解決できない場合は、Amazon のサポートに連絡しましょう。アカウント内部に何か不具合がある場合は、サポートスタッフが解決してくれる場合があります。

[Amazon 開発者専用お問い合わせ窓口](#)

リファレンス

Alexa の最新情報について

Echoデバイス及びAlexaの進化に伴い、Alexaのスキル開発環境も凄まじいスピードで進化を遂げています。Alexaのスキル開発に関する最新情報は下記のページから取得してください。

- [Alexa 開発者ポータルWebサイト](#)
- [Alexa Skills Kit 関連ドキュメント](#)
- [Alexa 開発者ブログ](#)
- [Alexa オフィシャルFacebookページ](#)
- [Alexa オフィシャルTwitterアカウント](#)

スキル開発に関するご質問

スキル開発にあたり、不明な点や疑問点がある場合は下記の Alexa技術者フォーラム または個別のお問い合わせフォーム に質問を投稿してください。専任の担当者がご質問に対応します。

- [Alexa 開発者フォーラム](#)
- [お問い合わせフォーム](#) (質問カテゴリでAlexaを選択してください)

教材に関するご意見やご要望など

この教材に関するご意見やご要望、誤植の報告などは下記のTwitterのハッシュタグを利用して投稿してください。品質の向上にご協力いただけすると幸いです。



#Alexaトレーニング

改定履歴（主な変更点）

- 2017.11.25 v1.0 初版（協力：クラスメソッド株式会社）
- 2018.04.12 v2.0 UIの変更に伴い、内容を大幅に改定
- 2018.05.16 v2.3 asciidocに移行（Alexa道場の動画リンクを追加）
- 2018.06.01 v2.5 Lambdaのポリシーテンプレート選択のスクリーンショットを更新
- 2018.06.03 v2.6 補足の追加（アカウントの問題の回避方法）
- 2018.06.13 v2.7 サンプルコードを SDK Version2 に更新
- 2018.06.18 v2.7.1 誤字修正、一部スクリーンショット差し替え
- 2018.07.10 v2.7.2 Alexaシミュレータのサンプル発話の入力でウェイクワードを省略
- 2018.10.28 v2.8.0 インテントの追加/スロットの演習を統合/発話のバリエーション/サンプルコードのASK CLI対応