



演習：カスタムスキルを作ろう

ASK Developer Marketing, Amazon Japan

Version 2.7.1, 2018/06/18

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. [課題1] 初めてのカスタムスキル	6
1.1. 対話モデルのデザイン	7
1.2. エンドポイントの開発	18
1.3. エンドポイントを登録する	28
1.4. テストする	30
2. [課題2] ビルトインスロットを使おう	36
2.1. 課題2のゴール	36
2.2. 課題2で作る対話モデル	36
2.3. Lambdaのコードを改良する	39
2.4. テストする	40
3. [課題3] カスタムスロットタイプを使おう	42
3.1. 課題3のゴール	42
3.2. 課題3で作る対話モデル	42
3.3. Lambdaのコードを改良する	46
3.4. テストする	47
4. [課題4] セッションアトリビュートを使おう	49
4.1. 課題4のゴール	49
4.2. 課題4で作る対話モデル	49
4.3. Lambdaのコードを改良する	50
4.4. テストする	55
補足	56
Alexaの開発者アカウントに関する問題	56
サンプルコード	63
リファレンス	74
Alexa の最新情報について	74
スキル開発に関するご質問	74
教材に関するご意見やご要望など	74
改定履歴（主な変更点）	75

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** [<https://alexa.desgin/jp-events>] 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演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** [<https://alexa.design/jp-alexadojo>] の収録動画をご覧ください。 **Alexa道場** は月2回のペースで定期的に開催し、スキル開発に関するテクニックの紹介や最新情報を届けています。第1回から第6回あたりまでがこの教材の内容をカバーしています。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](https://alexa.design/jp-alexadojo) [<https://alexa.design/jp-alexadojo>]

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。

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 [https://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](https://github.com/alexa/alexa-skills-kit-sdk-for-nodejs/blob/2.0.x/README.ja.md) [https://github.com/alexa/alexa-skills-kit-sdk-for-nodejs/blob/2.0.x/README.ja.md]

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](http://alexa.design/jp-text-v2) [http://alexa.design/jp-text-v2]

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。

- [SampleCodes_v2.zip](https://alexa.design/jp-samplecodes-v2) [https://alexa.design/jp-samplecodes-v2]

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。

尚、[サンプルコード](#)の内容はこのテキストの巻末にも掲載しています。

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** [<http://amzn.asia/4Blg18Y>]
- **Node.js超入門** [<http://amzn.asia/0O8YtY8>]
- **Node.js入門～サーバーサイドJavaScriptを根本から理解する** [<http://amzn.asia/0OdTtmK>]

1. [課題1] 初めてのカスタムスキル

この実習課題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。

例 1. 課題1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

1.1. 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。ユーザーに何を言われたら、どのインテントリクエストをエンドポイント（開発者のサーバー）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 [<https://alexa.design/jp-alexadojo003>]



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 [<https://developer.amazon.com/ja/docs/custom-skills/steps-to-build-a-custom-skill.html>]

1.1.1. 開発者コンソールへのログイン

1. 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。

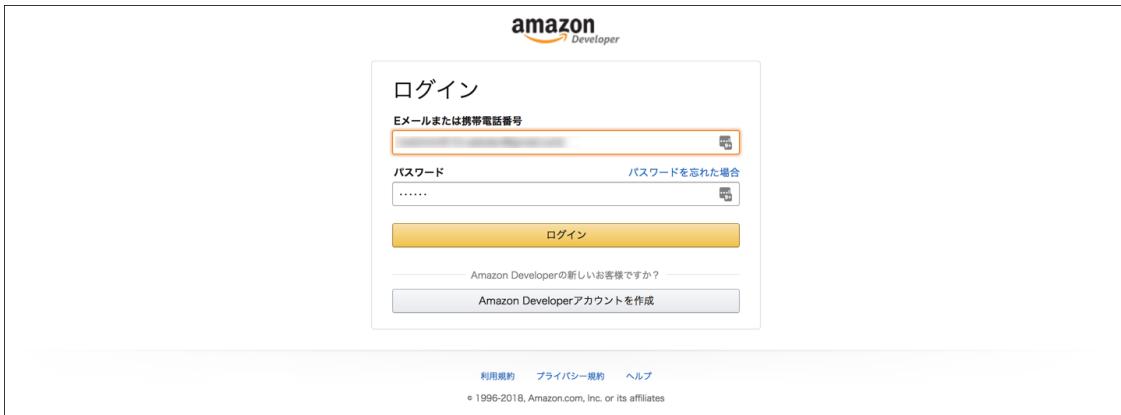
<https://developer.amazon.com/ja/> [<https://developer.amazon.com/ja/>]

2. 「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 the Amazon developer logo, a 'Developer Console' button (which is highlighted in red), a 'Sign In' link, and a search icon. Below the navigation, the text 'Amazon Developer Services and Technologies' is displayed. There are several service links arranged in a grid:

- Alexa**: 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**: 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
- Amazon API Store**: 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**: 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 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
- Amazon Web Services**: 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

3. 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。決して下の「Amazon Developerアカウントを作成」ボタンから開発者アカウントを作成しないでください。このボタンはUSの開発者アカウントを作成します。
- 特に支障がなければ普段Amazonでお買い物に利用しているアカウントをお使いください。
- 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る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？*

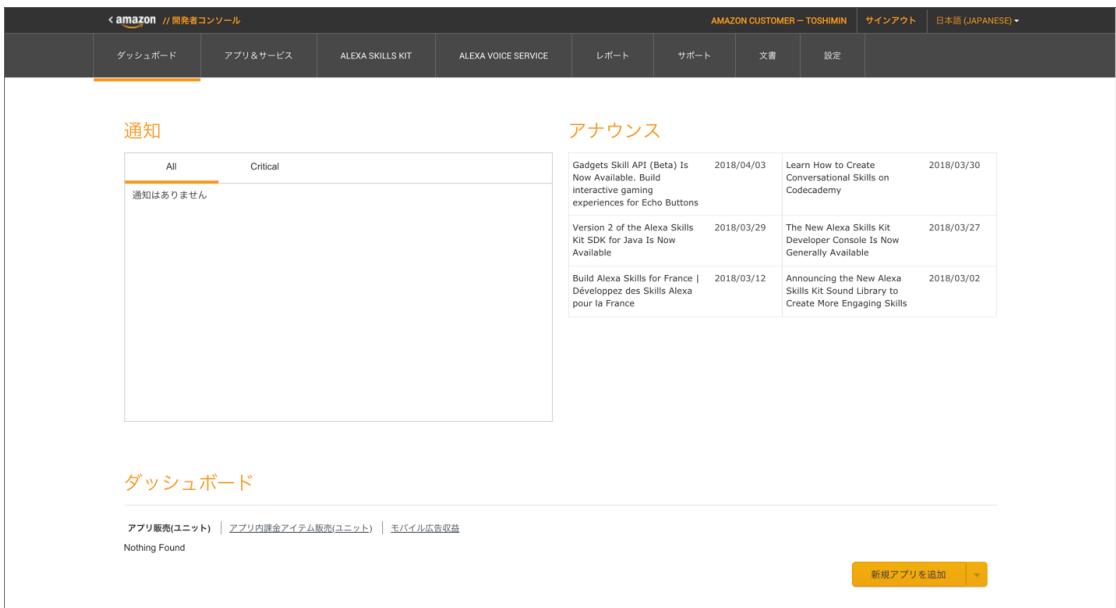
いいえ
 はい

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 Alexa Developer Console homepage. At the top, there's a navigation bar with links for Dashboard, Apps & Services, Alexa Skills Kit, Alexa Voice Service, Reports, Support, Documents, and Settings. The top right corner shows the user's name "AMAZON CUSTOMER – TOSHIMIN", a sign-out link, and a language selection dropdown set to "日本語 (JAPANESE)".

The main content area has two sections: "通知" (Notifications) and "アナウンス" (Announcements).

通知 (Notifications): A table with columns for Type (All/Critical), Title, Date, and Description.

All	Critical		
通知はありません			

アナウンス (Announcements): A table with columns for Title, Date, and Description.

	Date	Description
Gadgets Skill API (Beta) Is Now Available. Build interactive gaming experiences for Echo Buttons	2018/04/03	Learn How to Create Conversational Skills on Codecademy
Version 2 of the Alexa Skills Kit SDK for Java Is Now Available	2018/03/29	The New Alexa Skills Kit Developer Console Is Now Generally Available
Build Alexa Skills for France Développez des Skills Alexa pour la France	2018/03/12	Announcing the New Alexa Skills Kit Sound Library to Create More Engaging Skills

Below the announcements, there's a section titled "ダッシュボード" (Dashboard) with links for App Store (ユニット), App内課金アイテム販売(ユニット), and モバイル広告収益. It also shows a message "Nothing Found". A yellow button at the bottom right says "新規アプリを追加".

1.1.2. スキルの新規作成

- 「ALEXA SKILLS KIT」ボタンをクリックします。

The screenshot shows the Alexa Skills Kit developer console interface. At the top, there's a navigation bar with links like 'ダッシュボード', 'アプリ&サービス', 'ALEXA SKILLS KIT' (which is highlighted with a pink rectangle), 'ALEXA VOICE SERVICE', 'レポート', 'サポート', '文書', and '設定'. Below the navigation bar, there are two sections: '通知' (Notification) on the left and 'アナウンス' (Announcement) on the right. The announcement section contains a message about developers earning money for engaging skills, followed by a link to learn more. At the bottom of the announcement section, there are buttons for 'All' and 'Critical' notifications, and a timestamp '2018/04/03' along with a 'Learn How to Create' link.

- 「スキルの作成」ボタンをクリックします。

The screenshot shows the 'Skills' section of the Alexa Skills Kit developer console. It displays a table with columns for 'スキル名' (Skill Name), '言語' (Language), 'タイプ' (Type), '更新日' (Last Updated), 'ステータス' (Status), and 'アクション' (Actions). A green notification bar at the top indicates that a skill named 'コーヒーショップ' has been successfully deleted. In the center, there's a large blue button labeled 'Alexaスキル' (Alexa Skill) with a plus sign icon. Below it, another blue button labeled 'スキルの作成' (Create Skill) is highlighted with a pink rectangle. At the bottom of the page, there's a note about creating a new skill, a link to 'Alexa Skills Kit', and another 'スキルの作成' button.

- 「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、このスキル開発のプロジェクト名で、日本語でも英語でも構いません。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」を選択して「次へ」をクリックします。

The screenshot shows the first step of the 'Create New Skill' wizard, titled '新しいスキルを作成' (Create New Skill). It has a '次へ' (Next) button in the top right corner. The main area is titled 'スキル名' (Skill Name) with a text input field. Below it, a note says 'スキル作成時のデフォルト: 英語(米国)' (Default language for skill creation: English (US)). A dropdown menu shows language options: '英語(インド)', '英語(英国)', 'ドイツ語', and '日本語', with '日本語' currently selected. There's also a note at the bottom of the dropdown menu: 'フランス語(France)'.

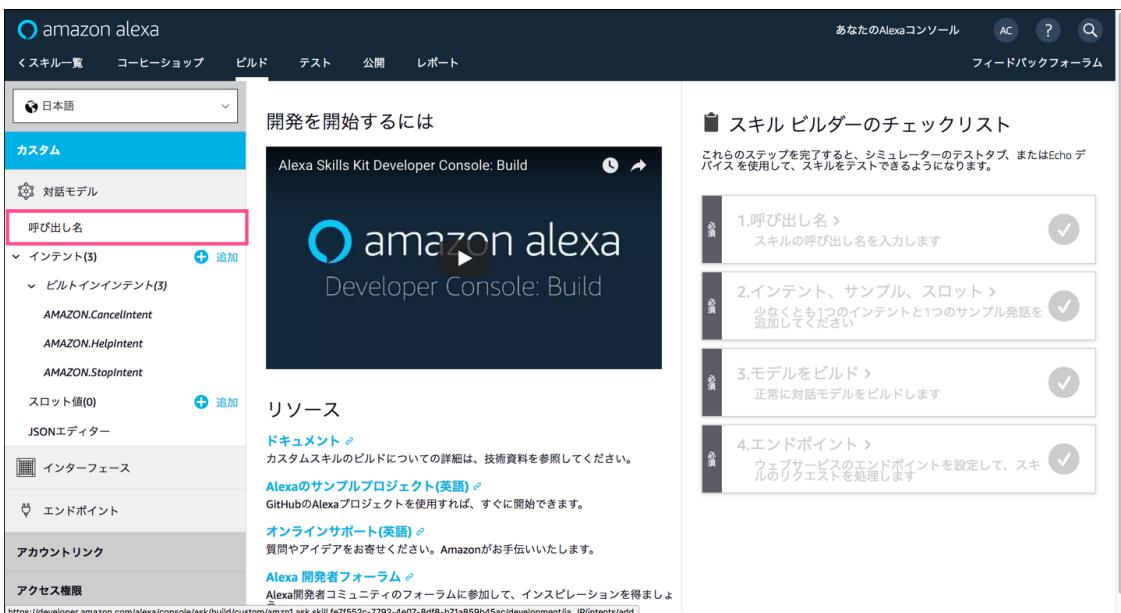
4. スキルに追加するモデルを選択します。この課題ではカスタムスキルを作成するので、カスタムの「選択」ボタンをクリックして「スキル作成」ボタンをクリックします。



5. スキルビルダーの画面が表示されます。

上のメニューで現在の作業ステップは「ビルド」であることを確認します。右側のチェックリストは、この画面で4つの作業ステップを完了すると、AlexaシミュレーターまたはEchoデバイスでテストができるようになることを表しています。

はじめに、左側の「対話モデル」タブにある「呼び出し名」をクリックします。



6. スキルの呼び出し名を入力します。

ここには、このスキルを呼び出すための名前を設定します。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」と入力しましょう。

そうすると、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

にスキルに話すことができるようになります。

入力が終わったら一度「モデルを保存」をクリックして保存します。

日本語

モデルを保存 モデルをビルド

呼び出し名

特定のカスタムスキルとの対話を開始するには、スキルの呼び出し名を発話します。たとえば、呼び出し名が「今日の星占い」の場合、次のように話しかけます。

ユーザー:アレクサ、今日の星占いを開いて双子座の運勢を占って

スキルの呼び出し名

コーヒーショップ

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」、「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ドするまで、スキルの呼び出し名の変更は反映されません。ビルド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つのサンプル発話の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

[<https://developer.amazon.com/ja/docs/custom-skills/steps-to-build-a-custom-skill.html>]



- カスタムスキルの呼び出し名を決定する

[<https://developer.amazon.com/ja/docs/custom-skills/choose-the-invocation-name-for-a-custom-skill.html>]

1.1.3. 対話モデルの作成

それでは、対話モデルのデザインに取りかかりましょう。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. 「対話モデル」タブの中のインテントの右にある「追加」をクリックしてカスタムインテントを追加します。

呼び出し名

特定のカスタムスキルとの対話を開始するには、スキルの呼び出し名を発話します。たとえば、呼び出し名が「今日の星占い」の場合、次のように話しかけます。

ユーチャー:アレクサ、今日の星占いを開いて双子座の運勢を占って

スキルの呼び出し名

コーヒーショップ

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」、「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つのサンプル発話の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](#)。

2. 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。

インテントを作成

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 [?](#)

OrderIntent

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 [?](#)

ビルトインインテントの使用についての[詳細はこちら](#)。

ビルトインを検索

ビルトイン名

説明

17/17個のビルトイン

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。



これを間違えると、エンドポイント側で正しい名前のインテントを取得できずエラーになります。

- 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を考えて入力します。ユーザーのコーヒーの注文の仕方は一つではありません。思いつく全てのフレーズをできるだけ多く入力してください。

サンプル発話の数は1インテントに対し最低6つ、理想は30と言われています。これは、サンプル発話がそのままAlexaの機械学習の教師モデルとなるため、サンプル発話が多いほど認識精度が向上するからです。



参考ドキュメント

- 音声インターフェース設計のベストプラクティス

[<https://developer.amazon.com/ja/docs/custom-skills/voice-design-best-practices-legacy.html>]

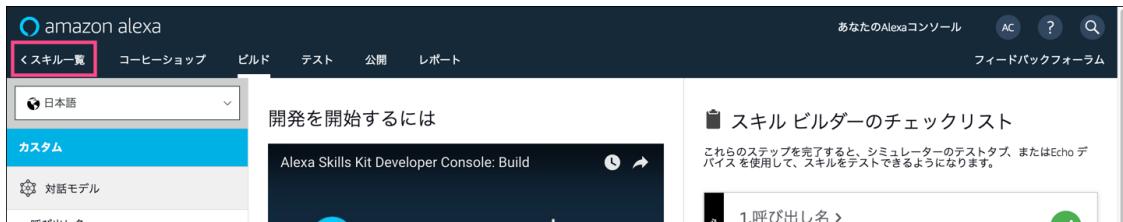
- インテントのサンプル発話の入力が完了したら「モデルを保存」をクリックし、最後に「モデルをビルト」ボタンをクリックします。モデルのビルトには1~2分かかる場合があります。

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。



「4. エンドポイント」はこの後のステップでAWS Lambdaを使ったプログラミングが終わった段階で設定します。ここでは一旦そのままにしておきます。

6. この後の処理で必要となるスキルIDの情報を取得するために、画面左上の「<スキル一覧」をクリックします。



7. スキル一覧に「コーヒーショップ」スキルがリストされています。スキル名の下に表示されている長い英数文字列を確認します。これはスキルIDと呼ばれるもので後のステップで使用します。文字列の左の小さなアイコンをクリックして文字列をコピーし、テキストエディターなどに貼り付けて保存しておいてください。

スキル名	言語	タイプ	更新日	ステータス
コーヒーショップ amzn1.ask.skill.fe7f552c-7792-4e07-8df8...	日本語	カスタム	2018-04-10	開発中

クリックして文字列をコピー

以上で、対話モデルのデザイン作業は終了です。ここまででAlexaはユーザーからの、「コーヒーが欲しい」と要求するさまざまな言い方を、エンドポイントのサーバー側がハンドリングしやすい **OrderIntent** というインテントリクエストに置き換え、サーバーに送信できるようになりました。

次に、インテントリクエスト **OrderIntent** を受け取る側、つまりエンドポイントのサーバーで処理をし、Alexaに応答を返すためのプログラムコードを書くステップに移りましょう。

1.2. 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。ここではAWSのLambdaというサービスを使い、Amazonが提供するAlexaスキル開発用のライブラリ [Alexa Skills Kit SDK for Node.js](#) [<https://github.com/alexa/alexa-skills-kit-sdk-for-nodejs>]を使ってプログラミングを行います。実習ではサンプルコードをコピーするだけですので安心して取り組んでください。



第4回 Alexa道場：Node.jsを使ってAlexaスキルを作ろう [<https://alexa.design/jp-alexadojo004>]



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](https://developer.amazon.com/ja/docs/custom-skills/host-a-custom-skill-as-an-aws-lambda-function.html)
[<https://developer.amazon.com/ja/docs/custom-skills/host-a-custom-skill-as-an-aws-lambda-function.html>]
- [Alexaから送信されたリクエストを処理する](https://developer.amazon.com/ja/docs/custom-skills/handle-requests-sent-by-alexa.html)
[<https://developer.amazon.com/ja/docs/custom-skills/handle-requests-sent-by-alexa.html>]

1.2.1. AWSマネージメントコンソールへのログイン

1. AWSポータルへアクセスします。

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

2. 「コンソールへログイン」をクリックします。

AWS Cloud9
ブラウザのみでコードを記述、実行、デバッグできるクラウドベースの統合開発環境
[詳細はこちる >](#)

リソース管理を始めましょう
[コンソールへサインイン »](#)

コンソールモバイルアプリ
iOS、Android デバイスでリソースチェックができます
[詳細・モバイル版ダウンロードはこちら »](#)

3. AWSアカウントIDを入力します。

aws

サインイン

AWS アカウントの E メールアドレス
IAM ユーザーとしてサインインするには、[アカウント ID](#) または [アカウントエイリアス](#) を入力してください。

次へ

—— AWS のご利用は初めてですか? ——

[新しい AWS アカウントの作成](#)

本番 Docker ワークLOADを実行する
Amazon EC2 Container Service

[今すぐ試す »](#)

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。必要であればIAMで開発用ユーザーを作成して開発を行ってください。



- 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。

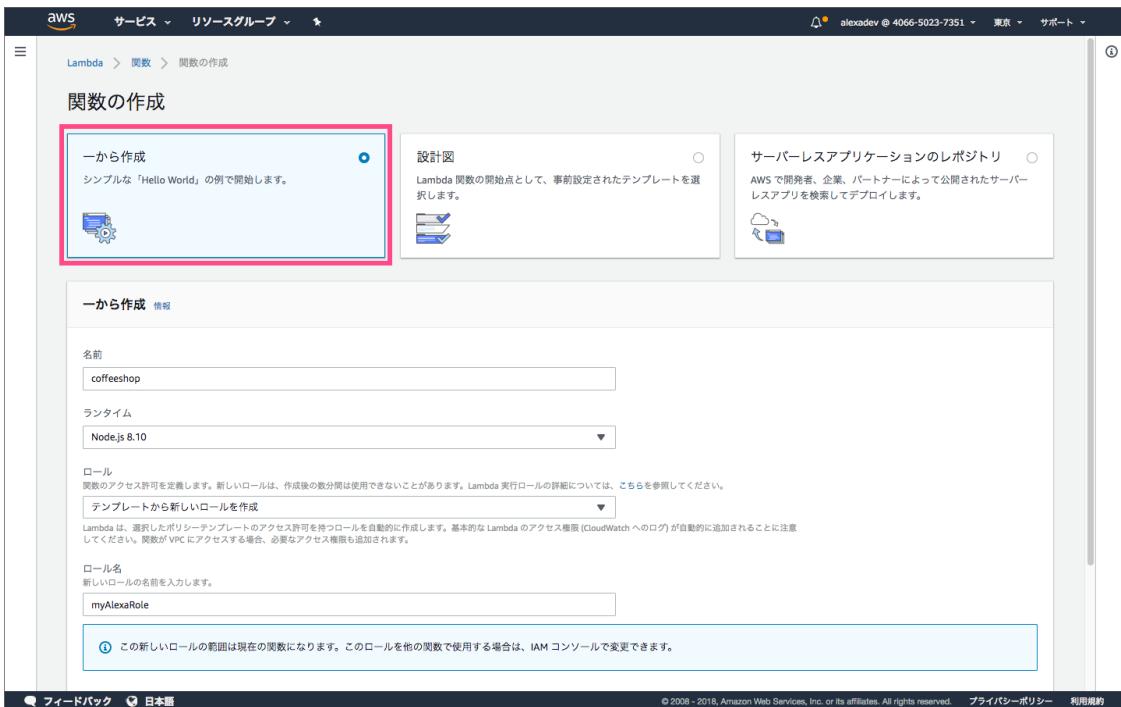


- 右上のリージョンを「アジアパシフィック（東京）」に変更し、「関数の作成」ボタンをクリックしてください。



- 「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つけてください。（例：AlexaRoleなど）



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

Q |

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

こちらを参照してください。

変更できます。

アクセス権限については、こちらを参照してください

9. 下図のようになつていれば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。

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。

Serviços > Lambda > 関数 > CoffeeShop

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

設定 条件 アクション 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。

Designer

トリガーの追加

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

CoffeeShop

Amazon CloudWatch Logs

Amazon DynamoDB

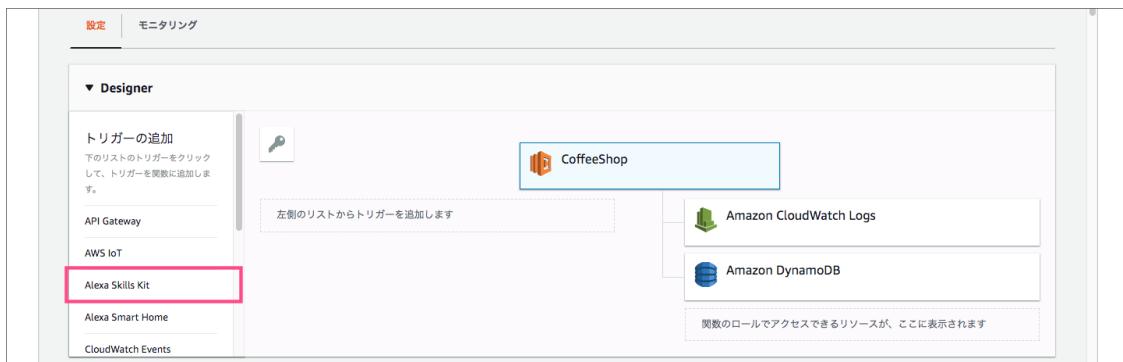
関数コード 情報

フィードバック 日本語

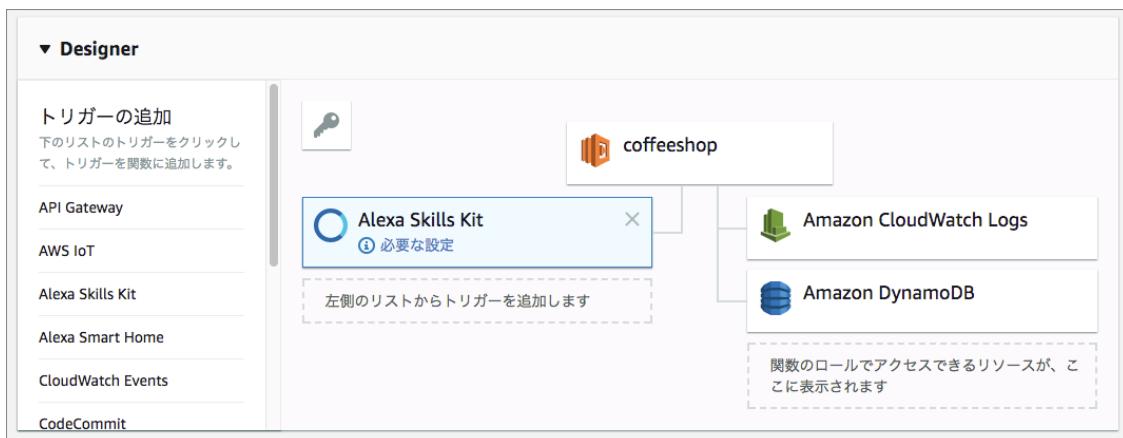
1.2.2. トリガーを追加する

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

Alexa Skills Kit ① 保存されていない変更 X

左側のリストからトリガーを追加します

coffeeshop

Amazon CloudWatch Logs
Amazon DynamoDB

1.2.3. 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

CoffeeShop X

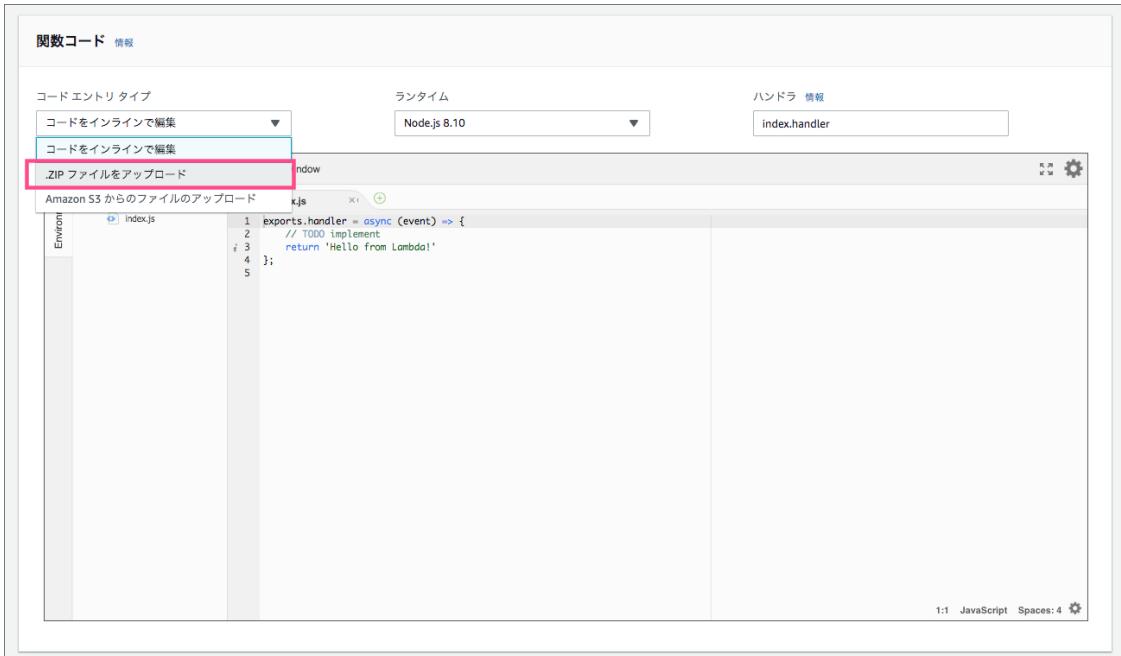
Alexa Skills Kit ① 必要な設定 X

左側のリストからトリガーを追加します

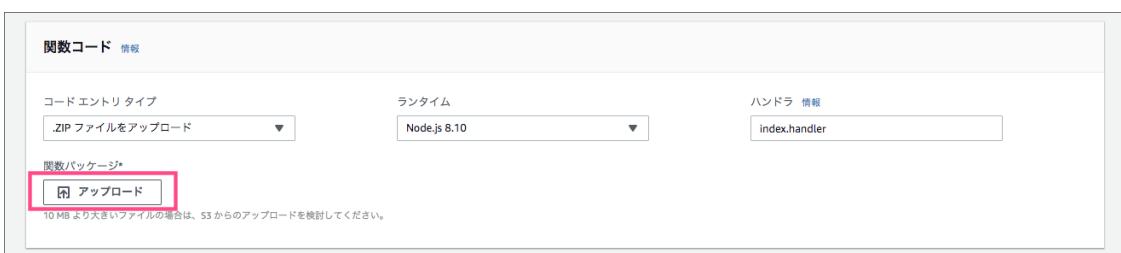
Amazon CloudWatch Logs
Amazon DynamoDB

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

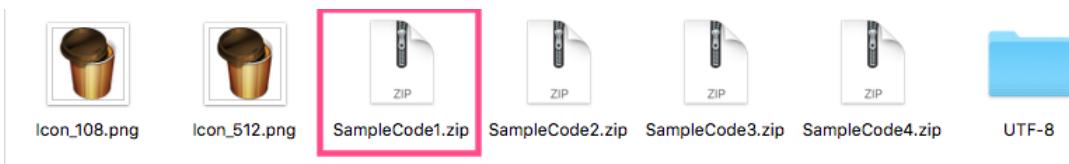
2. 画面の下方に関数のコードエディタが表示されます。左上のコードエントリタイプのプルダウンメニューから「.ZIPファイルをアップロード」を選択してください。



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、**SampleCode1.zip** ファイルを探し選択してください。



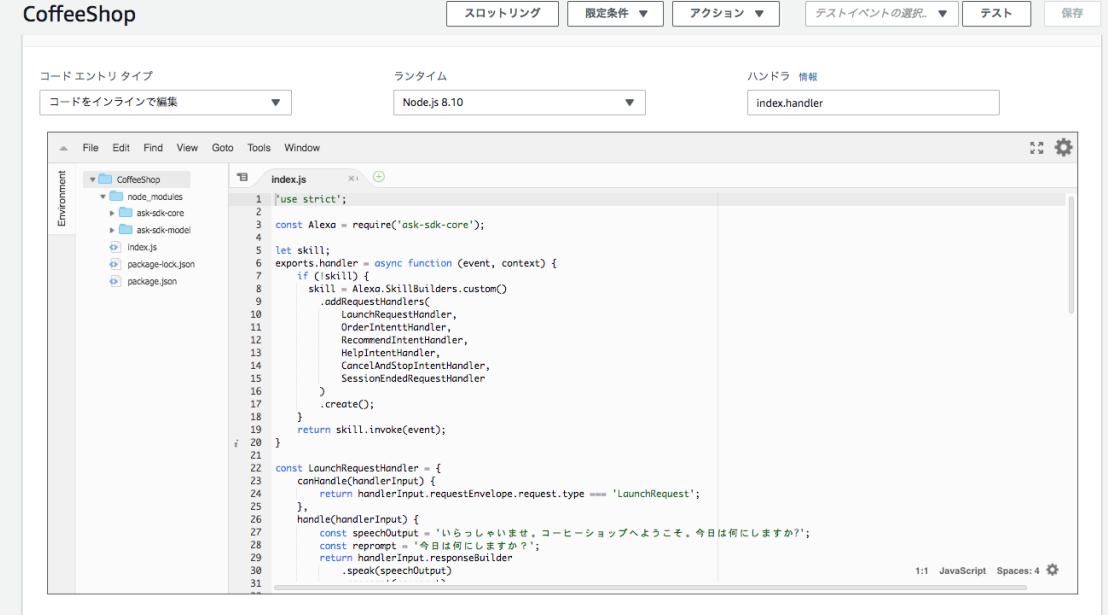
ソースコードとライブラリを含めたZIPファイルの作り方は、以下のSK SDK v2 for Node.js のドキュメントを参照してください。

[Setting Up The ASK SDK](https://github.com/alexa/alex-skills-kit-sdk-for-nodejs/wiki/%5BJapanese%5D-Setting-Up-The-ASK-SDK) [https://github.com/alexa/alex-skills-kit-sdk-for-nodejs/wiki/%5BJapanese%5D-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 **Sample1.js** のコードが **index.js** として読み込まれています。また、**ask-sdk-core** および **ask-sdk-model** のライブラリも同時にアップロードされていることが確認できます。



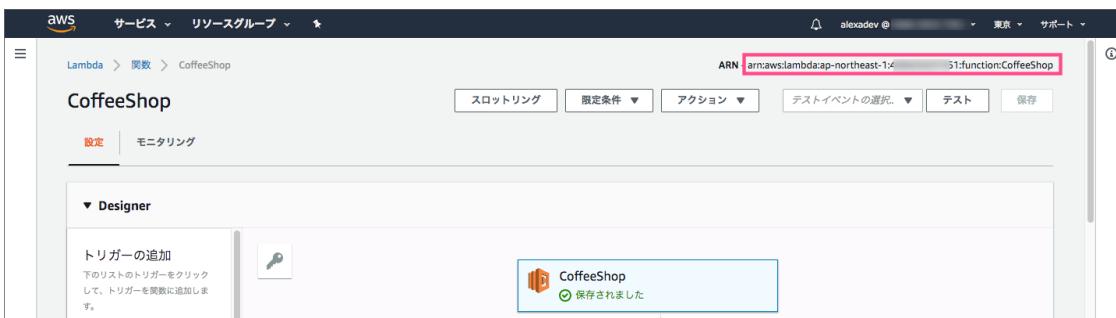
The screenshot shows the Alexa developer console's code editor interface. The title bar says "CoffeeShop". The top menu includes "スロットリング", "限定条件", "アクション", "テストイベントの選択", "テスト", and "保存". The left sidebar shows the project structure: "Environment" with "CoffeeShop" selected, containing "node_modules", "ask-sdk-core", "ask-sdk-model", "index.js", "package-lock.json", and "package.json". The main editor area displays the "index.js" file content:

```

    1 'use strict';
    2
    3 const Alexa = require('ask-sdk-core');
    4
    5 let skill;
    6 exports.handler = async function (event, context) {
    7   if (!skill) {
    8     skill = Alexa.SkillBuilders.custom()
    9       .addRequestHandlers(
    10         LaunchRequestHandler,
    11         OrderIntentHandler,
    12         RecommendIntentHandler,
    13         HelpIntentHandler,
    14         CancelAndStopIntentHandler,
    15         SessionEndedRequestHandler
    16       )
    17       .create();
    18   }
    19   return skill.invoke(event);
    20 }
    21 const LaunchRequestHandler = {
    22   canHandle(handlerInput) {
    23     return handlerInput.requestEnvelope.request.type === 'LaunchRequest';
    24   },
    25   handle(handlerInput) {
    26     const speechOutput = 'いらっしゃいませ。コーヒーショップへようこそ。今日は何にしますか？';
    27     const reprompt = '今日は何にしますか？';
    28     return handlerInput.responseBuilder
    29       .speak(speechOutput)
    30       .reprompt(reprompt)
    31       .build();
    32   }
  
```

The status bar at the bottom right indicates "1:1 JavaScript Spaces: 4".

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:XXXXXX: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。

1.3. 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。

- 「ビルド」タブを開き、左のメニューから「エンドポイント」をクリックします。

The screenshot shows the Alexa Skills Kit Developer Console interface. The top navigation bar includes links for 'スキル一覧' (Skill List), 'コピー・ショップ', 'ビルト', 'テスト', '公開', and 'レポート'. The left sidebar has a dropdown for language ('日本語') and sections for 'カスタム' (Custom), '対話モデル' (Dialog Model), '呼び出し名' (Invocation Name), 'インテント' (Intents), 'ビルトインインテント' (Built-in Intent), 'AMAZON.CancelIntent', 'AMAZON.HelpIntent', 'AMAZON.StopIntent', 'スロット値' (Slot Value), '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). The 'エンドポイント' section is highlighted with a pink box. The main content area displays the 'Build' tab of the Alexa Skills Kit Developer Console, showing the 'Developer Console: Build' interface with the Amazon Alexa logo. To the right, there is a 'スキルビルダーのチェックリスト' (Skill Builder Checklist) with four items, each marked with a green checkmark: 1.呼び出し名 (Skill invocation name), 2.インテント、サンプル、スロット (Intent, sample, slot), 3.モデルをビルト (Build dialog model), and 4.エンドポイント (Endpoint). Below the checklist, there are sections for 'リソース' (Resources), 'ドキュメント' (Documentation), 'Alexaのサンプルプロジェクト' (Alexa Sample Project), 'オンラインサポート' (Online Support), and 'Alexa開発者フォーラム' (Alexa Developer Forum).

- 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

エンドポイント

Alexaスキルとユーザーが対話すると、エンドポイントはPOSTリクエストを受け取ります。リクエストの本文には、サービスがロジックを実行し、JSON形式の応答を生成するために使用するパラメーターが含まれています。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](#)。ここで説明されている要件を満たしている限り、独自のHTTPS webサービスエンドポイント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 (必須)

スキルID (必須)

arn:aws:lambda:ap-northeast-1:406650237351:function:coffeeshop

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:<aws_account_id>:function:<lambda_name>

北米 (オプション)

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

ヨーロッパとインド (オプション)

arn:aws:lambda:eu-west-1:<aws_account_id>:function:<lambda_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

エンドポイント

Alexaスキルとユーザーが対話すると、エンドポイントはPOSTリクエストを受け取ります。リクエストの本文には、サービスがロジックを実行し、JSON形式の応答を生成するために使用するパラメーターが含まれています。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](#)。ここで説明されている要件を満たしている限り、独自のHTTPS webサービスエンドポイントをホスティングできます。

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。

1.4. 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** [<https://developer.amazon.com/ja/docs/devconsole/test-your-skill.html>]
- **ユーザーによるカスタムスキルの呼び出し**
[<https://developer.amazon.com/ja/docs/custom-skills/understanding-how-users-invoke-custom-skills.html>]

1.4.1. 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。



- 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。

JSON入力

```

1 - {
2 -   "version": "1.0",
3 -   "session": {
4 -     "new": true,
5 -     "sessionId": "amzn1.echo-api.session.dc554042-4dd0-40a7-b91c
6 -     "application": {
7 -       "applicationId": "amzn1.ask.skill.e2d259c3-d9dd-4be4-af
8 -     },
9 -     "user": {
10 -       "userId": "amzn1.ask.account.AG2527EYN5YV4SB0DR2B4FCVYZ
11 -     }
12 -   },
13 -   "context": {
14 -     "AudioPlayer": {
15 -       "playerActivity": "IDLE"
16 -     },
17 -     "Display": {},
18 -     "System": {
19 -       "application": {
20 -         "applicationId": "amzn1.ask.skill.e2d259c3-d9dd-4be4-af
21 -       },
22 -       "user": {
23 -         "userId": "amzn1.ask.account.AG2527EYN5YV4SB0DR2B4FCVYZ
24 -       },
25 -       "device": {
26 -         "deviceId": "amzn1.ask.device.AFYPORW837KPN7HRN00T1
27 -       },
28 -       "supportedInterfaces": {
29 -         "AudioPlayer": {}
30 -       },
31 -       "Display": {
32 -         "templateVersion": "1.0",
33 -         "markupVersion": "1.0"
34 -       }
35 -     }
36 -   }
37 - }

```

JSON出力

```

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。</speak>"
8 -       },
9 -       "shouldEndSession": true
10 -     },
11 -     "sessionAttributes": {}
12 -   }
13 - }

```

- 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す。
- 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。

1.4.2. (オプション) Echoデバイスでテストする

Alexaアプリのスキル一覧ページに開発中のスキル（DEV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。

1. Alexaアプリにログインし「スキル」タブをクリックしてください。

2. 「有効なスキル」をクリックしてください。

3. 「DEVスキル」のタブをクリックすると開発中のスキルのリストが表示されます。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。リストの中から、あなたが作成したスキルを探してください。

ホーム
再生中
ミュージック
リスト
リマインダー＆アラーム
スキル
スマートホーム
試してみよう!
設定
ヘルプとフィードバック
ASKさんではありませんか?サインアウト

有効なスキル
スキルの検索
すべてのスキル
90 有効なスキル
13 今更新されました
3 要注意です
⊕ さらに多くのスキル見つける
最近追加されたもの
すべてのスキル
今更新されました
要注意です
ペータスキー
DEV スキル

コーヒーショップ
Toshimi Hatanaka
★★★★★
"アレクサ、コーヒーショップを開いて"

コーヒーショップ応用編
devJP
"アレクサ、コーヒーショップを開いて"

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。

ホーム
再生中
ミュージック
リスト
リマインダー＆アラーム
スキル
スマートホーム
試してみよう!
設定
ヘルプとフィードバック
ASKさんではありませんか?サインアウト

コーヒーショップ
Toshimi Hatanaka
★★★★★
スキルを無効にする
話しかけ方の例
"アレクサ、コーヒーショップを開いて" "アレクサ、コーヒーショップでコーヒーを1つ注文して" "アレクサ、コーヒーショップでコーヒーをお願い"
このスキルについて
コーヒーを注文できるスキル
サポートされている言語
日本語 (JP)
スキルの詳細
• 呼び出し名：コーヒーショップ

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？

1.4.3. うまく動かない場合

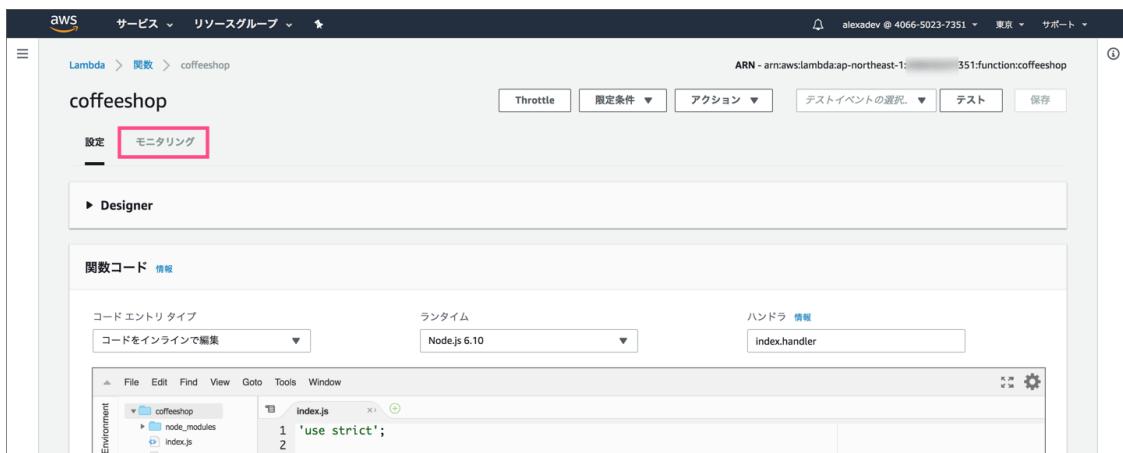
うまく動かない場合、様々な要因がありますが特に以下の設定を見直してみましょう。

- 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？

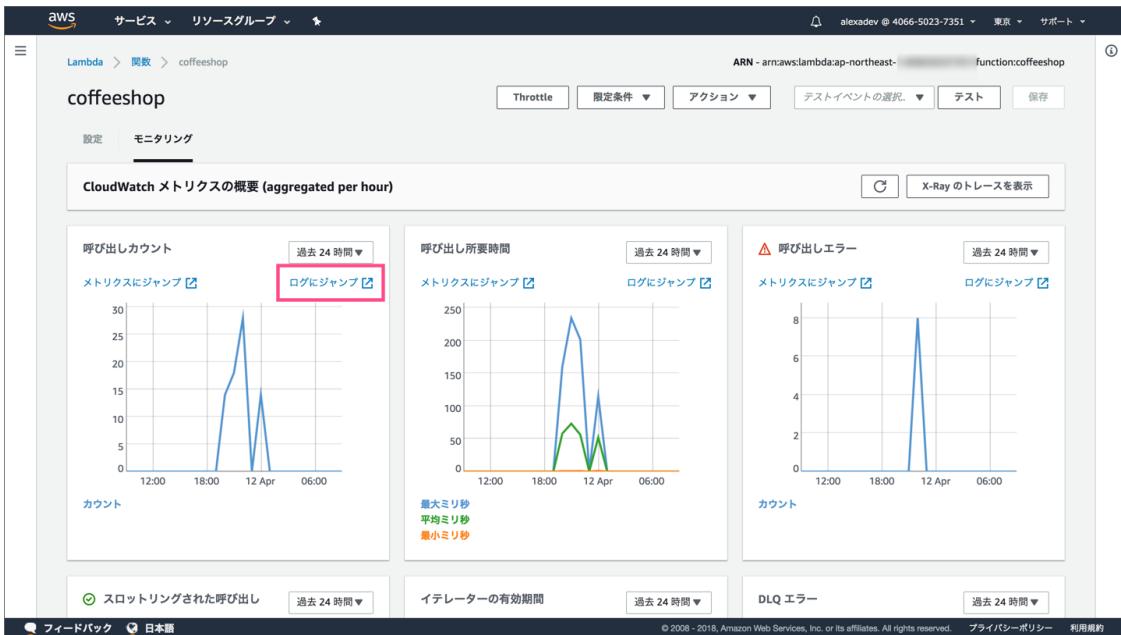
1.4.4. 「スキルからの応答に問題があります」？

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。

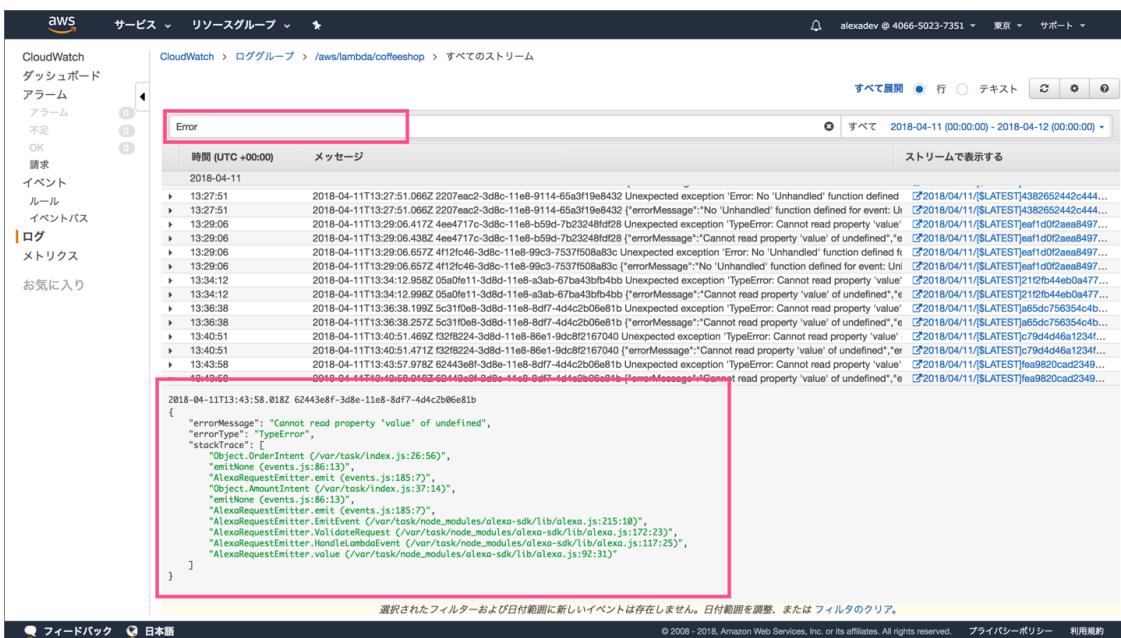
1. Lambda関数の編集画面を開き、「モニタリング」をクリックします。



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。



3. 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。



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！

次は課題2です。課題2では「スロット」を使ってコーヒーの数をスキルで受け取れるように改良します。先に進む前に講師の指示に従ってください。

2. [課題2] ビルトインスロットを使おう

2.1. 課題2のゴール

課題2では コーヒーの数 を受け取るようにスキルを改良します。

この課題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の種類を選択することができる。
- 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。

2.2. 課題2で作る対話モデル

例 2. 課題2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。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数字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 [\[https://developer.amazon.com/ja/docs/custom-skills/create-intents-utterances-and-slots.html\]](https://developer.amazon.com/ja/docs/custom-skills/create-intents-utterances-and-slots.html)
- スロットタイプリファレンス [\[https://developer.amazon.com/ja/docs/custom-skills/slot-type-reference.html\]](https://developer.amazon.com/ja/docs/custom-skills/slot-type-reference.html)

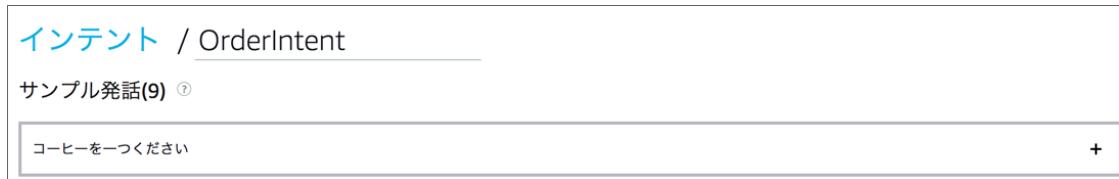


第5回 Alexa道場：スロットを使って発話内の可変文字列を取得しよう

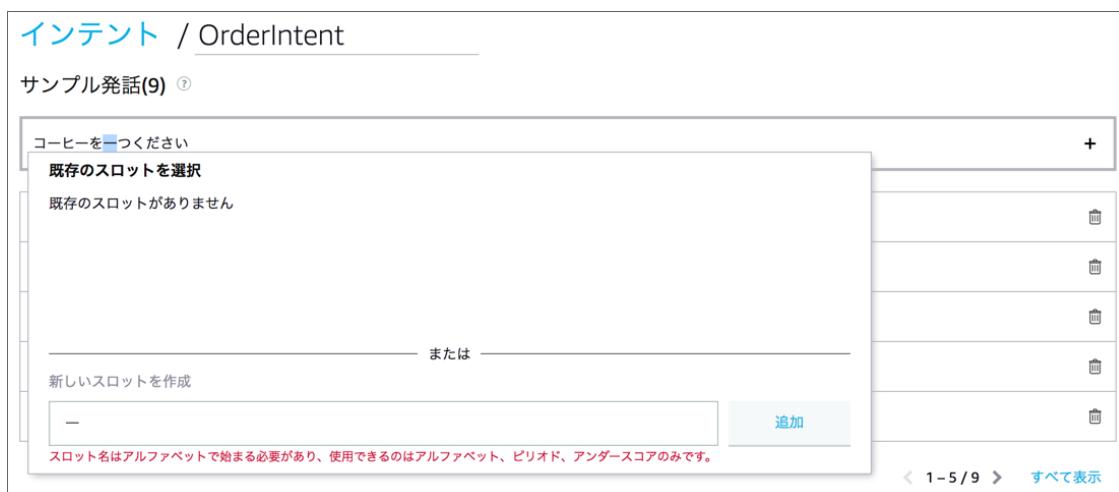
[<https://alexadesign.jp-alexadojo005>]

2.2.1. スロットを追加する

- 開発者コンソールに戻りスキルビルダーを開きます。
- インテント「**OrderIntent**」をクリックして、スロットを含むサンプル発話を追加します。まず「コーヒーを一つください」と入力してみましょう。



- スロットにしたい部分、この場合だと「一」（漢数字の1）の部分を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。



- 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。



- 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(9) ⑦

コーヒーを {amount} つください

+

コーヒーをください

■

6. 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) ⑦

順序 ⑦

名前 ⑦

スロットタイプ ⑦

アクション

^ 1

amount

2

新しいスロットを作成



ダイアログを編集
|
削除

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? ⑦



7. 次にもう一つ同じスロットamountを含むサンプル発話を追加します。「コーヒーを一つ注文して」と入力し、同じように漢字の「一」を選択します。今度は、既存のスロットを選択にamountが表示されるので、これを選択しクリックします。

インテント / OrderIntent

サンプル発話(10) ⑦

コーヒーを ■ 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

-

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。

< 1-5 / 10 > すべて表示

8. サンプル発話の入力中に、スロットを挿入したい部分でキーボードの { をタイプしてもスロット選択のポップアップが表示されます。好みの方法で、スロットを含むサンプル発話をどんどん追加してください。

インテント / OrderIntent

サンプル発話(11) ①

コーヒーを []
既存のスロットを選択
amount

コーヒーを
コーヒーを
コーヒーを
コーヒーを
コーヒーが

または 新しいスロットを作成
スロット名 追加

1-5 / 11 > すべて表示

9. 充分な数のサンプル発話を追加できたら「モデルを保存」し「モデルをビルト」をクリックしてください。

モデルを保存 モデルをビルト

インテント / OrderIntent

サンプル発話(14) 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

コーヒーが {amount} つ欲しいな
コーヒーを {amount} 杯よろしく
コーヒーを {amount} つお願い
コーヒーを {amount} つ注文して
コーヒーを {amount} つください

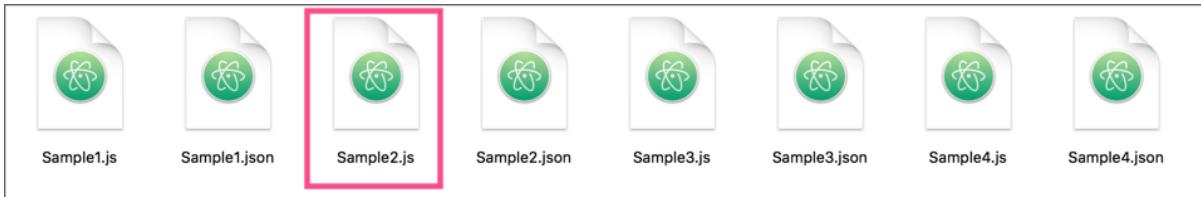


「～つ」、「～個」や「～杯」など数の数え方の単位も含め、あらゆるパターンを追加しましょう。ある程度はAlexaが自動で認識するようですが、何度もテストを繰り返して満足できる認識結果が得られるまで調整しましょう。

2.3. Lambdaのコードを改良する

次に、スロット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** をサンプルファイル **Sample2.js** の中身をコピーして貼り付けてください。



自分でコードを書いてみたい方は、以下のコードをヒントに、既存のコードに修正を加えてください。

リスト 3. スロット **amount** の値を取得し、**amount**の値を含めた応答を返すコード
(node.js)

```
1 var amount = handlerInput.requestEnvelope.request.intent.slots.amount.value;
2
3 const speechOutput = 'コーヒーを' + amount + '
4   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。';
5
6 return handlerInput.responseBuilder
7 .speak(speechOutput)
8 .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。

2.4. 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。

1. 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」という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。

The screenshot shows the Alexa Skills Kit interface. The top navigation bar includes links for 'スキル一覧', 'コーヒーショップ2', 'ビルト', 'テスト', '公開', '認定', 'レポート', and tabs for 'スキルI/O', 'Echo Showの画面', 'Echo Spotの画面', 'デバイスのログ'. The main area has tabs for 'Alexaシミュレーター', 'JSONエディタ', and '音声と語調'. A dropdown menu shows '日本語'. Below is a message box: '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。' On the right, there are two large code panes: 'JSON入力' and 'JSON出力'. The 'JSON入力' pane shows a complex JSON object representing an IntentRequest. The 'JSON出力' pane shows a simplified JSON response with an 'outputSpeech' field containing the spoken message.

- 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 **amount** に数字（下図の場合は1）が入ります。

This screenshot is similar to the previous one but highlights a specific part of the JSON input. A red box surrounds the 'amount' slot under the 'slots' key in the 'IntentRequest' object. An arrow points to the value '1' within the 'amount' slot object. The rest of the JSON structure and the output pane remain the same.

- 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。

3. [課題3] カスタムスロットタイプを使おう

ここまでのおしゃべりスキルでは、コーヒーしか注文できません。他のメニューも用意したいところですよね？そこで、カスタムスロットタイプを追加して、コーヒー以外にも「カフェラテ」「カプチーノ」なども注文できるようにしましょう。

3.1. 課題3のゴール

課題3ではカスタムスロットタイプを定義して任意のリストからスロットの値を取得します。

この課題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。

- カスタムスロットタイプを定義することができる。
- カスタムスロットタイプが設定されたスロットをサンプル発話内に追加できる。
- カスタムスロットタイプのスロットから送られたデータをLambda関数内で取得できる。

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](https://developer.amazon.com/ja/docs/custom-skills/create-and-edit-custom-slot-types.html) [https://developer.amazon.com/ja/docs/custom-skills/create-and-edit-custom-slot-types.html]

3.2. 課題3で作る対話モデル

例 3. 課題3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2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・・・」

課題2の対話モデルとほぼ同じですが、商品をコーヒー、カフェラテ、カプチーノ、エスプレッソなどから選択できるようになっています。

まずは、スロットでメニュー品目を取得できるよう、カスタムスロットタイプを追加します。

3.2.1. カスタムスロットタイプを追加する

- スキルビルダーを開き、左側のパネルからスロット値の「追加」ボタンをクリックします。

- 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。

スロットのタイプ / MenuList

スロット値(4) ②

このスロットタイプの新しい値を入力 +

値 ②	ID(オプション) ?	同義語(オプション) ②
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 >

4. 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(14) 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 +

コーヒーが {amount} つ欲しいな
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。

< 1 - 5 / 14 > すべて表示

5. 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(14) ?

このインテントの呼び出しに使われると思われる発話

+ 新規

コーヒーが {amount} つ欲しいな

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

イフアンドスロット(I) ?

1-5 / 14 すべて表示

6. 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 / OrderIntent

サンプル発話(14) ?

このインテントの呼び出しに使われると思われる発話

+ 新規

{menu} が {amount} つ欲しいな

コーヒーを {amount} 杯よろしく

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。

アクション

1-5 / 14 すべて表示

7. 他の既存のサンプル発話の「コーヒー」の部分を全てスロットmenuに置き換えましょう。手順は同様で既存のスロットの選択リストにmenuが追加されているはずです。

インテントスロット(2) ②

順序 ②	名前 ②	スロットタイプ ②	アクション
1	amount	AMAZON.NUMBER	ダイアログを編集 削除
2	menu	AMAZON.NUMBER	ダイアログを編集 削除
3	新しいスロットを作成	+ MenuList AMAZON.City AMAZON.DATE AMAZON.DURATION	ダイアログを編集 削除

この辺りを一度クリックするとmenuが表示されます

インテントの確認

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

amazon alexa

あなたのAlexaコンソール AC ? フィードバックフォーム

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
 - AMAZON.NUMBER
 - MenuList
 - JSONエディター

インテント / OrderIntent

サンプル発話(14) ②

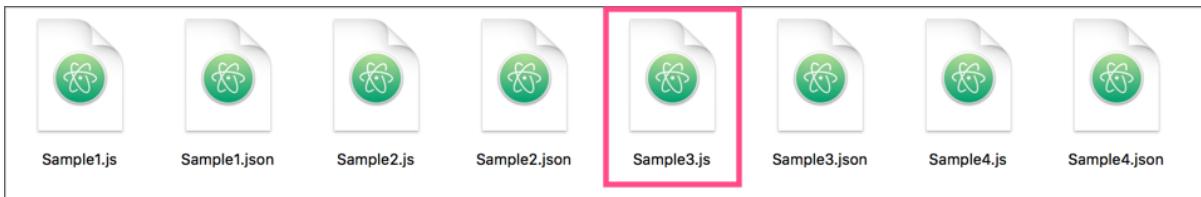
このインテントの呼び出しに使われると考えられる発話

- {menu} が {amount} つ欲しいな
- {menu} を {amount} 杯よろしく
- {menu} を {amount} つお願ひ
- {menu} を {amount} つ注文して
- {menu} を {amount} つください
- {menu} をください
- {menu} を注文して
- {menu} が欲しい
- {menu} にしようかな

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` をサンプルファイル [Sample3.js](#) の中身をコピーして貼り付けてください。



自分でコードを書いてみたい方は、以下のコードをヒントに、既存のコードに修正を加えてください。

リスト 4. スロット menu の値を取得し、menuの値を含めた応答を返すコード

```

1 var amount = handlerInput.requestEnvelope.request.intent.slots.amount.value;
2 var menu = handlerInput.requestEnvelope.request.intent.slots.menu.value;
3
4 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。';
5
6 return handlerInput.responseBuilder
7 .speak(speechOutput)
8 .getResponse();

```



ユーザーは欲しい品目を言ってくれない場合もあります。例えば「アレクサ、コーヒーショップで一つ注文して」のような場合です（このサンプル発話も追加しておく必要があります）。この場合、何を注文したいのかユーザーに聞き返す処理が必要で、そのコードは次のように書くことができます。この処理もコードに追加しておきましょう。

リスト 5. ユーザーが欲しい商品を言ってくれなかった場合の判別と、ユーザーに尋ねるコード

```

1 if (menu === undefined){
2     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？';
3     const reprompt = '何にしますか？';
4     return handlerInput.responseBuilder
5         .speak(speechOutput)
6         .reprompt(reprompt)
7         .getResponse();
8 }

```

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、カプチーノ

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。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。

The screenshot shows the Alexa Skills Kit Test Console interface. At the top, there are tabs for 'スキル一覧', 'コーヒーショップ3', 'ビルド', 'テスト', '公開', '認定', and 'レポート'. On the right, there are links for 'あなたのAlexaコンソール', 'A', '?', 'Q', and 'フィードバックフォーラム'. Below the tabs, there's a toggle switch labeled 'このスキルでは、テストは有効になっています' and several checkboxes: '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 has tabs for 'Alexaシミュレータ' (selected), 'JSONエディタ', and '音声と語調'. A dropdown menu shows '日本語'. Below it is a text input field with placeholder '入力またはマイクを長押しで発話' and a microphone icon. A button labeled '日本語' is also present. A message box displays the response: 'カブチーノを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。' To the right, there's a large blue play button icon. The bottom left shows a dark grey bar with '日本語' and a globe icon, and the bottom right shows copyright information: '© 2010-2018, Amazon.com, Inc. or its affiliates. All Rights Reserved. 無断複写・転載を禁じます。 利用規約 Alexa ブログ Alexa Skills Kit'.

スクリーンショット説明: このスクリーンショットは、Alexa Skills Kit の「コーヒーショップ3」スキルの「OrderIntent」に対する音声入力を示すものです。左側の JSON 入力欄には、Alexa が受け取った JSON リクエストが表示されています。右側の JSON 出力欄には、Alexa が返答した JSON レスポンスが表示されています。両方の JSON フィールド内に、赤枠で囲まれた部分があります。これは、音声入力で指定された「カブチーノ」の値が、JSON リクエストの「Slots」セクションと JSON レスポンスの「body」セクションの「value」フィールドに正しく反映されていることを示しています。

ここまでくると、ずいぶん実用に近いスキルになってきました。対話モデルやLambdaコードを微調整し完成度を高めましょう。

以上で課題3は終了です。お疲れ様でした！

4. [課題4] セッションアトリビュートを使う

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。

4.1. 課題4のゴール

課題3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。

この課題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 [\[https://developer.amazon.com/ja/docs/custom-skills/create-and-edit-custom-slot-types.html\]](https://developer.amazon.com/ja/docs/custom-skills/create-and-edit-custom-slot-types.html)

4.2. 課題4で作る対話モデル

例 4. 課題4の会話サンプル-1

人「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」

Alexa 「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？」

人「はい」

Alexa 「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った場合にも、それを尋ねる処理も加えましょう。

例 5. 課題4の会話サンプル-2

人「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」

Alexa 「コーヒーですね。いくつご用意しましょうか？」

人「二つお願い」

Alexa 「コーヒーを2つですね。お砂糖をおつけしますか？」

人「はい」

Alexa 「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。

4.3. Lambdaのコードを改良する

プログラミングに自信がない方は、サンプルコード [Sample4.js](#) の中身をコピーしてLambda関数の内容を置き換え、スキルビルダーの対話モデルのみを編集してください。



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 6. 変更前のコード

```
1 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。';
2 return handlerInput.responseBuilder
3 .speak(speechOutput)
4 .getResponse();
```

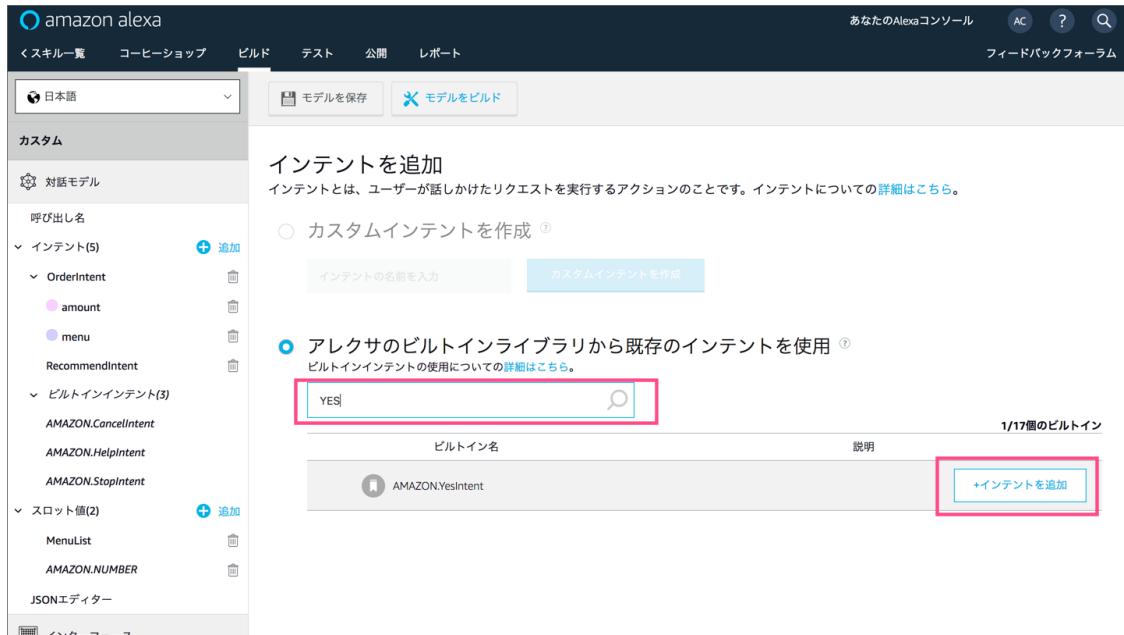
リスト 7. 変更後のコード

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

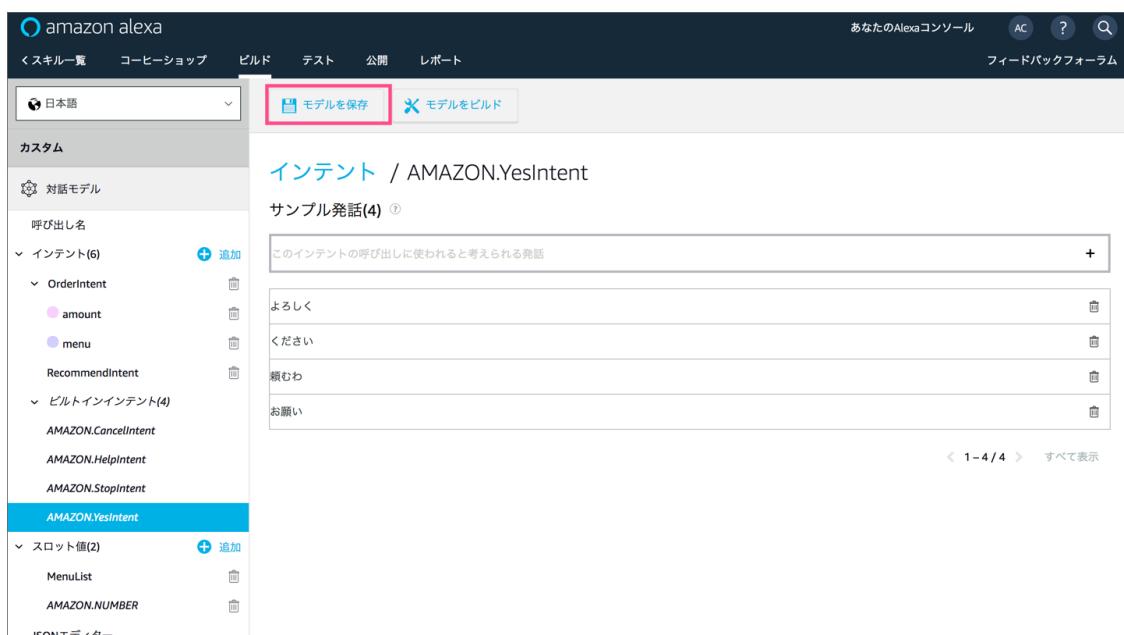
- 「お砂糖をおつけしますか？」を聞かれた後、ユーザーは「はい」か「いいえ」と言った応答をしてくるでしょう。これを受け取るインテントも用意しておきます。これには標準インテントの **AMAZON.YesIntent**、**AMAZON.NoIntent** を利用します。スキルビルダーに追加しておきましょう。
- スキルビルダーのインテント画面を開き「インテントを追加」ボタンをクリックします。

インテント名	発話	スロット	タイプ	アクション
OrderIntent	20	2	カスタム	編集 削除
RecommendIntent	7	-	カスタム	編集 削除
AMAZON.StopIntent	-	-	必須	編集
AMAZON.HelpIntent	-	-	必須	編集
AMAZON.CancelIntent	-	-	必須	編集

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。



5. 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。



6. 同様に、**AMAZON.NoIntent** も追加します。

インテント / AMAZON.NoIntent

サンプル発話(8) 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

男はブラックや
そんなんいらん
不要です
いりません
結構です

1-5 / 8 すべて表示

- もしお客様が数量を言ってくれなかった場合に、Alexaが数を聞き返す処理と、ユーザーが数量だけを話すスロットを含むインテント（AmountIntent）が必要になります。これらを追加しましょう。

リスト 8. ユーザーに数量を聞き返すコード

```

1 if(amount === undefined){
2     attributes.menu = menu;
3     handlerInput.attributesManager.setSessionAttributes(attributes);
4
5     const speechOutput = menu + 'ですね。おいくつご用意しましょうか？';
6     const reprompt = 'おいくつご用意しますか？';
7     return handlerInput.responseBuilder
8         .speak(speechOutput)
9         .reprompt(reprompt)
10        .getResponse();
11 }

```



図 1. ユーザーから数量の情報を受け取るインテントを追加する

8. 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 9. セッションアトリビュートのデータを保存するコード

```

1 attributes.menu = menu;
2 attributes.amount = amount;
3 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 10. セッションアトリビュートのデータを取り出すコード

```

1 const attributes = handlerInput.attributesManager.getSessionAttributes();
2 const amount = attributes.amount;
3 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。

4.4. 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 interface in test mode. The top navigation bar includes 'スキル一覧', 'コーヒーショップ4', 'ビルト', 'テスト' (selected), '公開', '認定', and 'レポート'. Below the navigation are checkboxes for 'スキルI/O', 'Echo Showの画面' (checked), 'Echo Spotの画面' (checked), and 'デバイスのログ' (unchecked). The main area has tabs for 'Alexaシミュレーター' (selected), 'JSONエディタ', and '音声と語調'. On the left, there's a language dropdown set to '日本語' and a microphone icon. The conversation log shows:

- User: コーヒーショップを開いてカフェラテを一つ注文して
- Bot: カフェラテを1つですね。お砂糖はおつけしますか？
- User: はい
- Bot: 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。

On the right, the 'スキルI/O' section displays JSON input and output logs. The JSON input is:

```

1 {
2   "version": "1.0",
3   "session": {
4     "new": false,
5     "sessionId": "amzn1.echo-api.session.1234567890abcdef",
6     "application": {
7       "applicationId": "amzn1.ask.skill.1234567890abcdef"
8     },
9     "attributes": {
10       "amount": "1",
11       "menu": "カフェラテ"
12     },
13     "user": {
14       "userId": "amzn1.ask.account.AB1234567890CDEF"
15     }
16   },
17   "context": {
18     "System": {
19       "application": {
20         "applicationId": "amzn1.ask.skill.1234567890abcdef"
21       }
22     },
23     "user": {
24       "userId": "amzn1.ask.account.AB1234567890CDEF"
25     }
26   }
}

```

The JSON output is:

```

1 {
2   "body": {
3     "version": "1.0",
4     "response": {
5       "outputSpeech": {
6         "type": "SSML",
7         "ssml": "<speak>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。</speak>"
8       }
9     },
10    "sessionAttributes": {
11      "amount": "1",
12      "menu": "カフェラテ"
13    },
14    "userAgent": "ask-node/2.0.7 Node/v8.11.3"
15  }
16 }

```

以上で課題4は終了です。お疲れ様でした！

補足

Alexaの開発者アカウントに関する問題

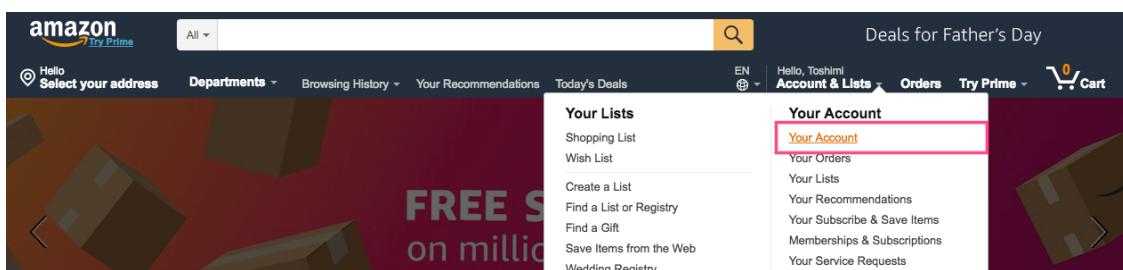
Alexaのスキル開発で使用する開発者アカウントは、Amazon Echoをセットアップした時に使用したアカウントと同じものを使用することを推奨します。作成したスキルを、同じアカウントでセットアップされたお手持ちのEchoデバイスすぐにテストできた方が便利だからです。

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> [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 'Your Account' section of the Amazon Alexa website. At the top, there's a navigation bar with links like 'Hello', 'Select your address', 'Departments', 'Browsing History', 'Your Recommendations', 'Today's Deals', 'EN', 'Hello, Toshimi', 'Account & Lists', 'Orders', 'Try Prime', and a shopping cart icon. Below the navigation, it says 'Toshimi's Amazon' and offers 'AMAZON PRIME Try Prime View benefits' and 'AUDIBLE AUDIOBOOKS Try Audible Get 2 free audiobooks'. The main area has several buttons: 'Your Orders' (Track, return, or buy things again), 'Login & security' (Edit login, name, and mobile number), 'Prime' (View benefits and payment settings), 'Your Addresses' (Edit addresses for orders and gifts), 'Payment options' (Edit or add payment methods), and 'Gift cards' (View balance or redeem a card). The 'Login & security' button is highlighted with a red box.

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 'Sign in' page of the Amazon website. It features the Amazon logo at the top, followed by a 'Sign in' button. Below it is a 'Switch accounts' link and a placeholder for a user profile picture with the name 'Toshimi Hatanaka'. There are fields for 'Password' and 'Forgot your password?' with a 'Sign in' button below them. A checkbox for 'Keep me signed in.' is also present.

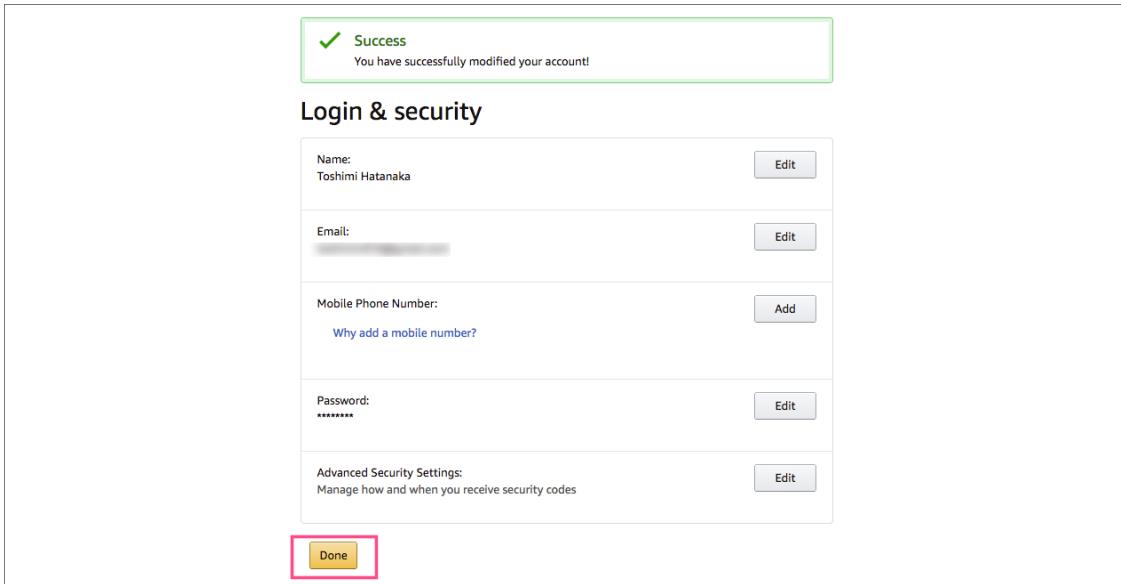
4. Password:の項目の「Edit」ボタンをクリックします。

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

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 to 'Use the form below to change the password for your Amazon account'. It includes fields for 'Current password' (input field), 'New password' (input field with a password strength indicator), 'Reenter new password' (input field with a password strength indicator), and a 'Save changes' button at the bottom.

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 (with a link to add one), Password (*****), and Advanced Security Settings. Each field has an 'Edit' button. At the bottom is a yellow 'Done' button.

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>

[<http://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 has fields for 'Eメールまたは携帯電話番号' and 'パスワード', both with 'Forgot password?' links. Below them is a large yellow 'ログイン' (Login) button. At the bottom, there are links for 'Amazon Developerの新しいお客様ですか？' and 'Amazon Developerアカウントを作成'.

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 *

10. 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。

1. <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

ログイン

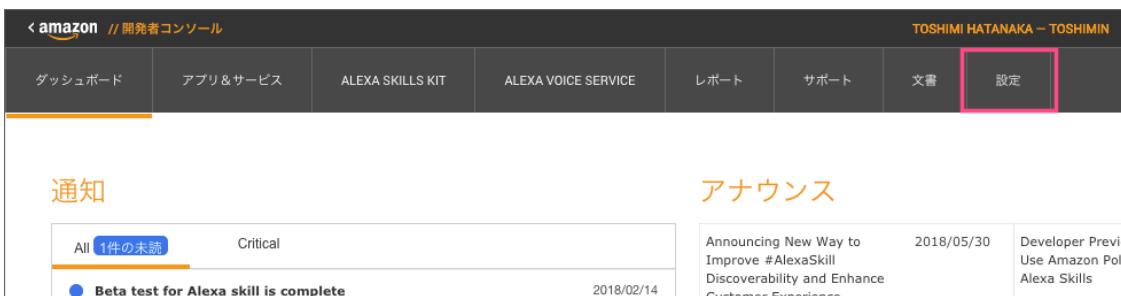
Amazon Developerの新しいお客様ですか？

Amazon Developerアカウントを作成

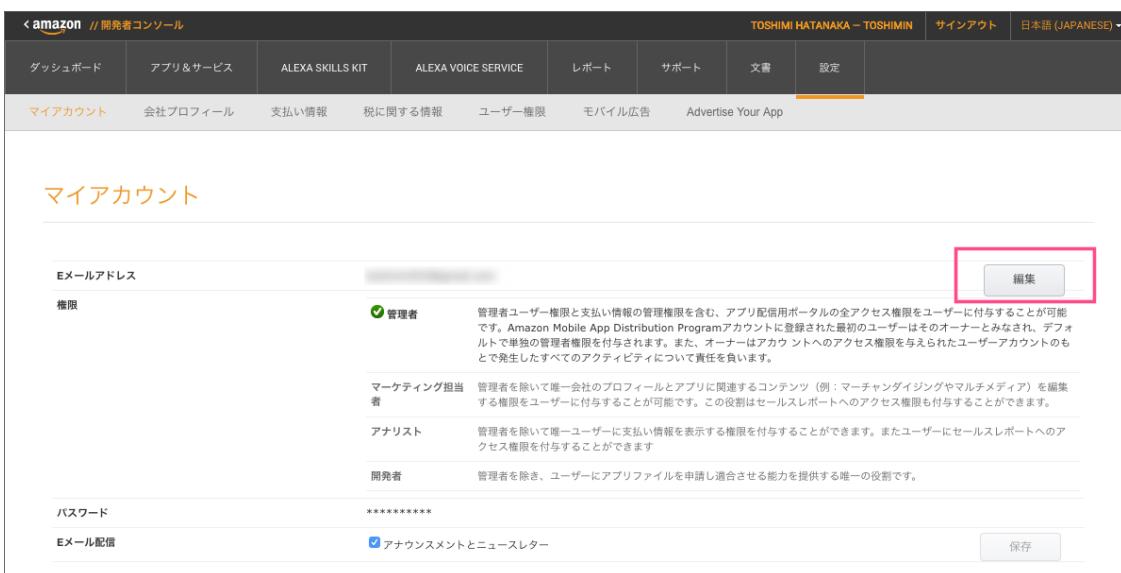
2. ログインしたら「DeveloperPortal」のリンクをクリックします。



3. 「設定」をクリックします。



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。



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 (highlighted with a red box)

携帯電話番号: 追加
携帯電話番号を追加する理由

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:

新しいEメールアドレス:

新しいEメールアドレスの再入力:

現在のパスワード:

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

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

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

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。

The screenshot shows a registration form titled '申請' (Application). At the top, there are three tabs: '1. プロフィール情報' (Profile Information), '2. App Distribution Agreement', and '3. 支払い' (Payment). A note below the tabs says '＊は必須項目を示しています。' (Fields marked with * are required). The '国/リージョン' (Country/Region) field is set to '日本' (Japan). Below it are fields for '名' (First Name) and '姓' (Last Name), both with placeholder text 'XXXXXX'. The 'Eメールアドレス' (Email Address) field contains 'XXXXXX@XXXXXX'. The '電話番号' (Phone Number) field has '11111111111111' entered, with a note 'e.g. 212-555-1212, +44 0161 715 3369'. The 'Fax番号' (Fax Number) field is empty. The '開発者名' (Developer Name) field has 'Amazon.co.jp' and 'Amazon.co.jp' entered. The '開発者名(ふりがな)' (Developer Name Katakana) field has 'Amazon' and 'Amazon' entered. The '開発者概要' (Developer Summary) field is a large text area with a blue border, currently empty. The '住所1' (Address Line 1) field has 'XXXXXX' entered. A note at the bottom left of the form says '最大文字数: 4000, 残り: 4000'.

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 [<https://developer.amazon.com/ja/support/contact-us>]

サンプルコード

リスト 11. Sample1.js

```

1 const Alexa = require('ask-sdk-core');
2
3 const LaunchRequestHandler = {
4     canHandle(handlerInput) {
5         return handlerInput.requestEnvelope.request.type === 'LaunchRequest';
6     },
7     handle(handlerInput) {
8         const speechOutput = 'いらっしゃいませ。コーヒーショップへようこそ。今日は何にしますか?';
9         const reprompt = '今日は何にしますか?';
10        return handlerInput.responseBuilder
11            .speak(speechOutput)
12            .reprompt(reprompt)
13            .getResponse();
14    }
15 };
16
17 const OrderIntentHandler = {
18     canHandle(handlerInput) {
19         return handlerInput.requestEnvelope.request.type === 'IntentRequest'
20             && handlerInput.requestEnvelope.request.intent.name === 'OrderIntent';
21     },
22     handle(handlerInput) {
23         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円でいいですよ。またの御利用をお待ちしております。';
24         return handlerInput.responseBuilder
25             .speak(speechOutput)
26             .getResponse();
27    }
28 };
29
30 const RecommendIntentHandler = {
31     canHandle(handlerInput) {
32         return handlerInput.requestEnvelope.request.type === 'IntentRequest'
33             && handlerInput.requestEnvelope.request.intent.name === 'RecommendIntent';
34    },
35     handle(handlerInput) {
36         const speechOutput = '今日は甘さスッキリのカフェラテがおすすめです。';
37         return handlerInput.responseBuilder
38             .speak(speechOutput)
39             .getResponse();
40    }
41 };
42
43 const HelpIntentHandler = {
44     canHandle(handlerInput) {
45         return handlerInput.requestEnvelope.request.type === 'IntentRequest'
46             && handlerInput.requestEnvelope.request.intent.name === 'AMAZON.HelpIntent';
47    },
48     handle(handlerInput) {
49         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?';
50         const reprompt = '今日は何にしますか?';
51         return handlerInput.responseBuilder
52             .speak(speechOutput)
53             .reprompt(reprompt)
54             .getResponse();

```

```
55     }
56 };
57
58 const CancelAndStopIntentHandler = {
59   canHandle(handlerInput) {
60     return handlerInput.requestEnvelope.request.type === 'IntentRequest'
61       && (handlerInput.requestEnvelope.request.intent.name === 'AMAZON.CancelIntent'
62           || handlerInput.requestEnvelope.request.intent.name === 'AMAZON.StopIntent');
63   },
64   handle(handlerInput) {
65     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。';
66     return handlerInput.responseBuilder
67       .speak(speechOutput)
68       .getResponse();
69   }
70 };
71
72 const SessionEndedRequestHandler = {
73   canHandle(handlerInput) {
74     return handlerInput.requestEnvelope.request.type === 'SessionEndedRequest';
75   },
76   handle(handlerInput) {
77     return handlerInput.responseBuilder.getResponse();
78   }
79 };
80
81 const UnhandledIntentHandler = {
82   canHandle(handlerInput) {
83     return true;
84   },
85   handle(handlerInput) {
86     const message = "いずれのandlerにも該当しないリクエストがきた場合に処理されます。";
87     return handlerInput.responseBuilder
88       .speak(message)
89       .getResponse()
90   }
91 };
92
93 const ErrorHandler = {
94   canHandle () {
95     return true
96   },
97   handle (handlerInput, error) {
98     console.log(`Error handled: ${error.message}`)
99     const message = "すみません、なんだかうまく行かないようです。もう一度お試しください。";
100    return handlerInput.responseBuilder
101      .speak(message)
102      .getResponse()
103   }
104 };
105
106 const skillBuilder = Alexa.SkillBuilders.custom();
107 exports.handler = skillBuilder
108   .addRequestHandlers(
109     LaunchRequestHandler,
110     OrderIntentHandler,
111     RecommendIntentHandler,
112     HelpIntentHandler,
113     CancelAndStopIntentHandler,
114     SessionEndedRequestHandler,
115     UnhandledIntentHandler
116   )
117   .addErrorHandlers(ErrorHandler)
```

```
118    .lambda();
```

リスト 12. Sample2.js

```
1 const Alexa = require('ask-sdk-core');
2
3 const LaunchRequestHandler = {
4     canHandle(handlerInput) {
5         return handlerInput.requestEnvelope.request.type === 'LaunchRequest';
6     },
7     handle(handlerInput) {
8         const speechOutput = 'いらっしゃいませ。コーヒーショップへようこそ。今日は何にしますか?';
9         const reprompt = '今日は何にしますか?';
10        return handlerInput.responseBuilder
11            .speak(speechOutput)
12            .reprompt(reprompt)
13            .getResponse();
14    }
15 };
16
17 const OrderIntentHandler = {
18     canHandle(handlerInput) {
19         return handlerInput.requestEnvelope.request.type === 'IntentRequest'
20             && handlerInput.requestEnvelope.request.intent.name === 'OrderIntent';
21     },
22     handle(handlerInput) {
23         var amount = handlerInput.requestEnvelope.request.intent.slots.amount.value;
24         if (amount === undefined){
25             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円でいいですよ。またの御利用をお待ちしております。';
26
27             return handlerInput.responseBuilder
28                 .speak(speechOutput)
29                 .getResponse();
30
31         } else {
32             const speechOutput = 'コーヒーを' + amount +
'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。';
33
34             return handlerInput.responseBuilder
35                 .speak(speechOutput)
36                 .getResponse();
37         }
38     }
39 };
40
41 const RecommendIntentHandler = {
42     canHandle(handlerInput) {
43         return handlerInput.requestEnvelope.request.type === 'IntentRequest'
44             && handlerInput.requestEnvelope.request.intent.name === 'RecommendIntent';
45     },
46     handle(handlerInput) {
47         const speechOutput = '今日は甘さスッキリのカフェラテがおすすめです。';
48         return handlerInput.responseBuilder
49             .speak(speechOutput)
50             .getResponse();
51     }
52 };
53
54 const HelpIntentHandler = {
55     canHandle(handlerInput) {
56         return handlerInput.requestEnvelope.request.type === 'IntentRequest'
57             && handlerInput.requestEnvelope.request.intent.name === 'AMAZON.HelpIntent';
58 }
```

```
58 },
59     handle(handlerInput) {
60         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?';
61         const reprompt = '今日は何にしますか?';
62         return handlerInput.responseBuilder
63             .speak(speechOutput)
64             .reprompt(reprompt)
65             .getResponse();
66     }
67 };
68
69 const CancelAndStopIntentHandler = {
70     canHandle(handlerInput) {
71         return handlerInput.requestEnvelope.request.type === 'IntentRequest'
72             && (handlerInput.requestEnvelope.request.intent.name === 'AMAZON.CancelIntent'
73                 || handlerInput.requestEnvelope.request.intent.name === 'AMAZON.StopIntent');
74     },
75     handle(handlerInput) {
76         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。';
77         return handlerInput.responseBuilder
78             .speak(speechOutput)
79             .getResponse();
80     }
81 };
82
83 const SessionEndedRequestHandler = {
84     canHandle(handlerInput) {
85         return handlerInput.requestEnvelope.request.type === 'SessionEndedRequest';
86     },
87     handle(handlerInput) {
88         return handlerInput.responseBuilder.getResponse();
89     }
90 };
91
92 const ErrorHandler = {
93     canHandle () {
94         return true;
95     },
96     handle (handlerInput, error) {
97         console.log(`Error handled: ${error.message}`);
98         const message = "すみません、うまく聞き取れませんでした。もう一度言ってください。";
99         return handlerInput.responseBuilder
100             .speak(message)
101             .reprompt(message)
102             .getResponse()
103     }
104 };
105
106 const skillBuilder = Alexa.SkillBuilders.custom();
107 exports.handler = skillBuilder
108     .addRequestHandlers(
109         LaunchRequestHandler,
110         OrderIntentHandler,
111         RecommendIntentHandler,
112         HelpIntentHandler,
113         CancelAndStopIntentHandler,
114         SessionEndedRequestHandler
115     )
116     .addErrorHandlers(ErrorHandler)
117     .lambda();
```

リスト 13. Sample3.js

```
1 const Alexa = require('ask-sdk-core');
2
3 const LaunchRequestHandler = {
4     canHandle(handlerInput) {
5         return handlerInput.requestEnvelope.request.type === 'LaunchRequest';
6     },
7     handle(handlerInput) {
8         const speechOutput = 'いらっしゃいませ。コーヒーショップへようこそ。今日は何にしますか?';
9         const reprompt = '今日は何にしますか?';
10        return handlerInput.responseBuilder
11            .speak(speechOutput)
12            .reprompt(reprompt)
13            .getResponse();
14    }
15 };
16
17 const OrderIntentHandler = {
18     canHandle(handlerInput) {
19         return handlerInput.requestEnvelope.request.type === 'IntentRequest'
20             && handlerInput.requestEnvelope.request.intent.name === 'OrderIntent';
21    },
22    handle(handlerInput) {
23        var amount = handlerInput.requestEnvelope.request.intent.slots.amount.value;
24        var menu = handlerInput.requestEnvelope.request.intent.slots.menu.value;
25        if (menu == undefined){
26            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?';
27            const reprompt = '何にしますか?';
28            return handlerInput.responseBuilder
29                .speak(speechOutput)
30                .reprompt(reprompt)
31                .getResponse();
32        } else if (amount == undefined){
33            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。';
34            return handlerInput.responseBuilder
35                .speak(speechOutput)
36                .getResponse();
37        } else {
38            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あります。';
39            return handlerInput.responseBuilder
40                .speak(speechOutput)
41                .getResponse();
42        }
43    }
44 };
45
46 const RecommendIntentHandler = {
47     canHandle(handlerInput) {
48         return handlerInput.requestEnvelope.request.type === 'IntentRequest'
49             && handlerInput.requestEnvelope.request.intent.name === 'RecommendIntent';
50    },
51    handle(handlerInput) {
52        const speechOutput = '今日は甘さスッキリのカフェラテがおすすめです。';
53        return handlerInput.responseBuilder
54            .speak(speechOutput)
55            .getResponse();
56    }
}
```

```
57 };
58
59 const HelpIntentHandler = {
60   canHandle(handlerInput) {
61     return handlerInput.requestEnvelope.request.type === 'IntentRequest'
62       && handlerInput.requestEnvelope.request.intent.name === 'AMAZON.HelpIntent';
63   },
64   handle(handlerInput) {
65     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?';
66     const reprompt = '今日は何にしますか?';
67     return handlerInput.responseBuilder
68       .speak(speechOutput)
69       .reprompt(reprompt)
70       .getResponse();
71   }
72 };
73
74 const CancelAndStopIntentHandler = {
75   canHandle(handlerInput) {
76     return handlerInput.requestEnvelope.request.type === 'IntentRequest'
77       && (handlerInput.requestEnvelope.request.intent.name === 'AMAZON.CancelIntent'
78           || handlerInput.requestEnvelope.request.intent.name === 'AMAZON.StopIntent');
79   },
80   handle(handlerInput) {
81     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。';
82     return handlerInput.responseBuilder
83       .speak(speechOutput)
84       .getResponse();
85   }
86 };
87
88 const SessionEndedRequestHandler = {
89   canHandle(handlerInput) {
90     return handlerInput.requestEnvelope.request.type === 'SessionEndedRequest';
91   },
92   handle(handlerInput) {
93     return handlerInput.responseBuilder.getResponse();
94   }
95 };
96
97 const ErrorHandler = {
98   canHandle () {
99     return true;
100   },
101   handle (handlerInput, error) {
102     console.log(`Error handled: ${error.message}`);
103     const message = "すみません、うまく聞き取れませんでした。もう一度言ってください。";
104     return handlerInput.responseBuilder
105       .speak(message)
106       .reprompt(message)
107       .getResponse()
108   }
109 };
110
111 const skillBuilder = Alexa.SkillBuilders.custom();
112 exports.handler = skillBuilder
113   .addRequestHandlers(
114     LaunchRequestHandler,
115     OrderIntentHandler,
116     RecommendIntentHandler,
117     HelpIntentHandler,
118     CancelAndStopIntentHandler,
```

```
119     SessionEndedRequestHandler  
120     )  
121     .addErrorHandlers(ErrorHandler)  
122     .lambda();
```

リスト 14. Sample4.js

```

1 const Alexa = require('ask-sdk-core');
2
3 const LaunchRequestHandler = {
4     canHandle(handlerInput) {
5         return handlerInput.requestEnvelope.request.type === 'LaunchRequest';
6     },
7     handle(handlerInput) {
8         const speechOutput = 'いらっしゃいませ。コーヒーショップへようこそ。今日は何にしますか?';
9         const reprompt = '今日は何にしますか?';
10        return handlerInput.responseBuilder
11            .speak(speechOutput)
12            .reprompt(reprompt)
13            .getResponse();
14    }
15 };
16
17 const OrderIntentHandler = {
18     canHandle(handlerInput) {
19         return handlerInput.requestEnvelope.request.type === 'IntentRequest'
20             && (handlerInput.requestEnvelope.request.intent.name === 'OrderIntent' ||
21                 handlerInput.requestEnvelope.request.intent.name === 'AmountIntent');
22    },
23    handle(handlerInput) {
24        const attributes = handlerInput.attributesManager.getSessionAttributes();
25        let amount = attributes.amount;
26        let menu = attributes.menu;
27        if (amount === undefined){
28            if (handlerInput.requestEnvelope.request.intent.slots.amount) {
29                amount = handlerInput.requestEnvelope.request.intent.slots.amount.value;
30            }
31        }
32        if (menu === undefined){
33            if (handlerInput.requestEnvelope.request.intent.slots.menu) {
34                menu = handlerInput.requestEnvelope.request.intent.slots.menu.value;
35            }
36        }
37
38        if(menu === undefined){
39            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?';
40            const reprompt = '何にしますか?';
41            return handlerInput.responseBuilder
42                .speak(speechOutput)
43                .reprompt(reprompt)
44                .getResponse();
45        }else if(amount === undefined){
46            attributes.menu = menu;
47            handlerInput.attributesManager.setSessionAttributes(attributes);
48
49            const speechOutput = menu + 'ですね。おいくつご用意しましょうか?';
50            const reprompt = 'おいくつご用意しますか?';
51            return handlerInput.responseBuilder
52                .speak(speechOutput)
53                .reprompt(reprompt)
54                .getResponse();
55        }else{
56            attributes.menu = menu;
57            attributes.amount = amount;
58            handlerInput.attributesManager.setSessionAttributes(attributes);
59        }
}

```

```
60         const speechOutput = menu + 'を' + amount + 'つですね。お砂糖はおつけしますか?';
61         const reprompt = 'お砂糖はおつけしますか?';
62         return handlerInput.responseBuilder
63             .speak(speechOutput)
64             .reprompt(reprompt)
65             .getResponse();
66     }
67 }
68 };
69
70 const YesIntentHandler = {
71     canHandle(handlerInput) {
72         return handlerInput.requestEnvelope.request.type === 'IntentRequest'
73             && handlerInput.requestEnvelope.request.intent.name === 'AMAZON.YesIntent';
74     },
75     handle(handlerInput) {
76         const attributes = handlerInput.attributesManager.getSessionAttributes();
77         const amount = attributes.amount;
78         const menu = attributes.menu;
79
80         const speechOutput = menu + 'を' + amount +
81             '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。';
82         return handlerInput.responseBuilder
83             .speak(speechOutput)
84             .getResponse();
85     }
86 };
87
88 const NoIntentHandler = {
89     canHandle(handlerInput) {
90         return handlerInput.requestEnvelope.request.type === 'IntentRequest'
91             && handlerInput.requestEnvelope.request.intent.name === 'AMAZON.NoIntent';
92     },
93     handle(handlerInput) {
94         const attributes = handlerInput.attributesManager.getSessionAttributes();
95         const amount = attributes.amount;
96         const menu = attributes.menu;
97
98         const speechOutput = menu + 'を' + amount +
99             'つ。お砂糖なしでご用意いたします。ご利用ありがとうございました。';
100        return handlerInput.responseBuilder
101            .speak(speechOutput)
102            .getResponse();
103    }
104 };
105
106 const RecommendIntentHandler = {
107     canHandle(handlerInput) {
108         return handlerInput.requestEnvelope.request.type === 'IntentRequest'
109             && handlerInput.requestEnvelope.request.intent.name === 'RecommendIntent';
110     },
111     handle(handlerInput) {
112         const speechOutput = '今日は甘さスッキリのカフェラテがおすすめです。';
113         return handlerInput.responseBuilder
114             .speak(speechOutput)
115             .getResponse();
116    }
117 };
118
119 const HelpIntentHandler = {
120     canHandle(handlerInput) {
121         return handlerInput.requestEnvelope.request.type === 'IntentRequest'
122             && handlerInput.requestEnvelope.request.intent.name === 'AMAZON.HelpIntent';
```

```

121 },
122 handle(handlerInput) {
123   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?';
124   const reprompt = '今日は何にしますか?';
125   return handlerInput.responseBuilder
126     .speak(speechOutput)
127     .reprompt(reprompt)
128     .getResponse();
129 }
130 };
131
132 const CancelAndStopIntentHandler = {
133   canHandle(handlerInput) {
134     return handlerInput.requestEnvelope.request.type === 'IntentRequest'
135       && (handlerInput.requestEnvelope.request.intent.name === 'AMAZON.CancelIntent'
136           || handlerInput.requestEnvelope.request.intent.name === 'AMAZON.StopIntent');
137   },
138   handle(handlerInput) {
139     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。';
140     return handlerInput.responseBuilder
141       .speak(speechOutput)
142       .getResponse();
143   }
144 };
145
146 const SessionEndedRequestHandler = {
147   canHandle(handlerInput) {
148     return handlerInput.requestEnvelope.request.type === 'SessionEndedRequest';
149   },
150   handle(handlerInput) {
151     return handlerInput.responseBuilder.getResponse();
152   }
153 };
154
155 const ErrorHandler = {
156   canHandle () {
157     return true;
158   },
159   handle (handlerInput, error) {
160     console.log(`Error handled: ${error.message}`);
161     const message = "すみません、うまく聞き取れませんでした。もう一度言ってください。";
162     return handlerInput.responseBuilder
163       .speak(message)
164       .reprompt(message)
165       .getResponse()
166   }
167 };
168
169 const skillBuilder = Alexa.SkillBuilders.custom();
170 exports.handler = skillBuilder
171   .addRequestHandlers(
172     LaunchRequestHandler,
173     OrderIntentHandler,
174     RecommendIntentHandler,
175     YesIntentHandler,
176     NoIntentHandler,
177     HelpIntentHandler,
178     CancelAndStopIntentHandler,
179     SessionEndedRequestHandler
180   )
181   .addErrorHandlers(ErrorHandler)
182   .lambda();

```

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 [<https://developer.amazon.com/alexa>]
- Alexa Skills Kit 関連ドキュメント [<https://developer.amazon.com/ja/docs/ask-overviews/build-skills-with-the-alexa-skills-kit.html>]
- Alexa 開発者ブログ [<https://developer.amazon.com/ja/blogs/alexa/tag/japan>]
- Alexa オフィシャルFacebookページ [<https://www.facebook.com/AlexaDevsJP/>]
- Alexa オフィシャルTwitterアカウント [<https://twitter.com/AlexaDevsJP>]

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 [<http://alexa.design/jp-forum>]
- お問い合わせフォーム [<https://developer.amazon.com/ja/support/contact-us>] (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 [<https://twitter.com/search?src=typd&q=%23Alexa%E3%83%88%E3%83%AC%E3%83%BC%E3%83%8B%E3%83%B3%E3%82%B0>]

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