

Alexaスキルを作ってみよう

マネージメントサービス株式会社 風間玲央

Alexaスキルとは

Alexaは、Amazon Echo、Echo Dotをはじめとする数多くのデバイスで中枢的な働きをするAmazonの音声サービスです。

<https://developer.amazon.com/ja/alexa-skills-kit>

⇒スマートスピーカーやスマートフォンに話しかけると、色んなことをしてくれる音声サービス！

Amazon Echo

Amazon Echoは、音声による操作で、常にハンズフリーで利用でき、いつでも反応します。お客様が部屋のさまざまなところから声をかけ、各種の情報や音楽の再生、ニュース、天気などの情報を求めると、Alexaが直ぐに対応します。



Amazon Echo Dot

Echo Dotは声で制御するハンズフリーの端末で、Amazon Echoと同じ遠隔音声認識を使っています。Echo Dotには、スピーカーが内蔵されているほか、Bluetoothや付属のオーディオケーブルでお手持ちのスピーカーに接続することもできます。



今回作るスキル

好きな言葉(+です)を話しかけると、Alexaがその言葉をそのまま返します。

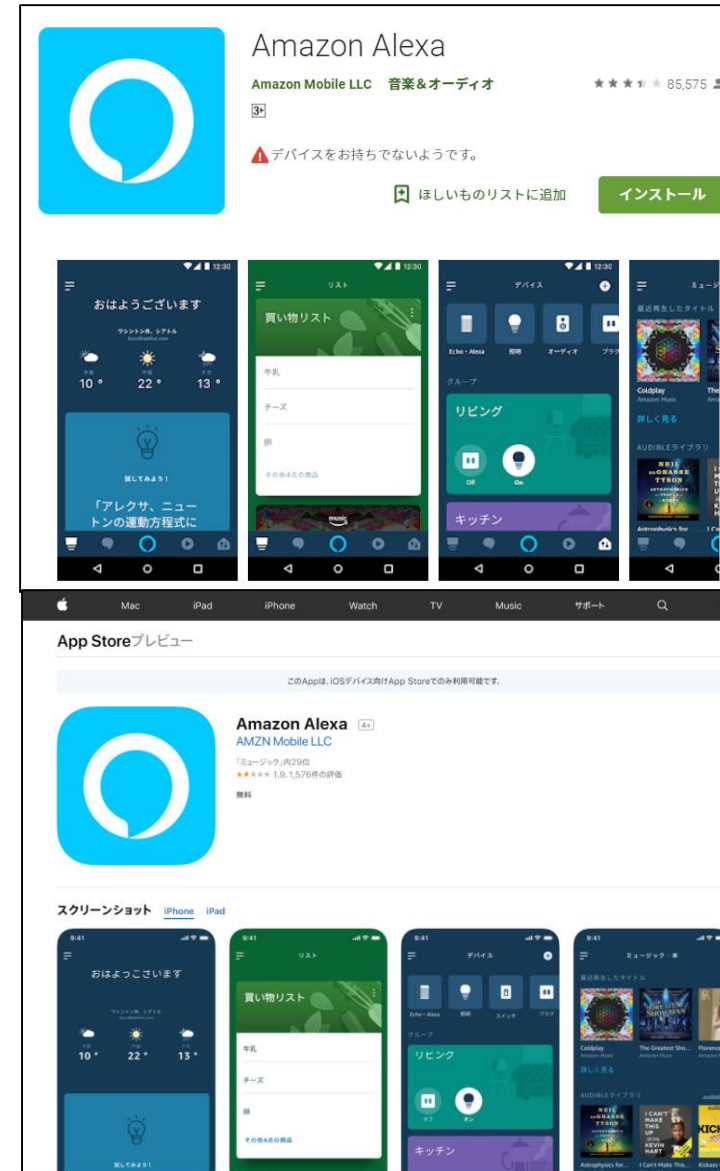
目的: AWSを使って、簡単に音声サービスが作れることを体験する



事前準備

- 必ず用意: PCとAmazon Developerアカウント
(Amazonのショッピングサイトのアカウントを登録できます。)⇒登録方法はスライド5～11へ
- あると良い1: Visual Studio Code
(<https://azure.microsoft.com/ja-jp/products/visual-studio-code/>)
- (検索エンジンで「Visual Studio Code」で検索してインストール)
- あると良い2: スマートフォンとAlexaアプリ(アプリストアから「Amazon Alexa」で検索してインストール)
- あると便利かも: マイク付きのイヤホン・ヘッドホン
(Alexaとのやり取りで使用。なくてもテストはブラウザ上でテキストのやり取りでできます。)

Alexaアプリ

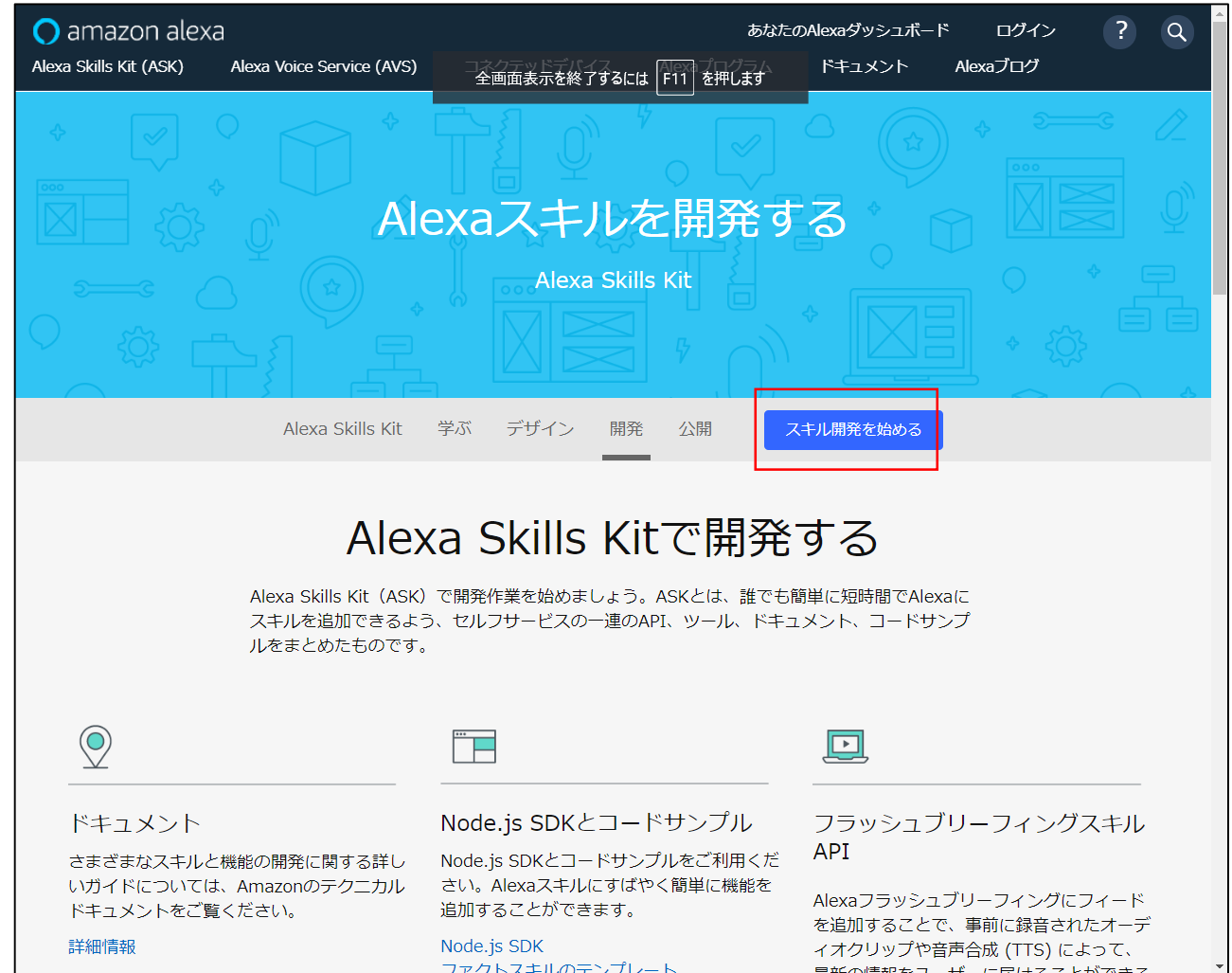


Amazon Developerアカウントの登録

スキル開発を始める

Alexaスキルを開発する
(<https://developer.amazon.com/ja/alexa-skills-kit/build>)にアクセスし、「スキル開発を始める」をクリックします。

「Alexaスキルを開発する」で検索すると出てきます。



ログイン

Amazonアカウントでログインします。

アカウントがなければ、新しくAmazonアカウントを作成します。

amazon alexa

ログイン

Eメールまたは携帯電話番号
[Redacted] .com

パスワード [パスワードを忘れた場合](#)
.....

ログイン

続行することで、Amazonの[利用規約](#)および[プライバシー規約](#)に同意するものとみなされます。

☐ ログインしたままにする [詳細](#) ▼

Amazonの新しいお客様ですか?

Amazonアカウントを作成

[利用規約](#) [プライバシー規約](#) [ヘルプ](#)

© 1996-2019, Amazon.com, Inc. or its affiliates

プロフィール入力

ログイン後、開発者としてのプロフィール情報を入力します。メールアドレス、電話番号、住所等は個人のもので会社のもので、アカウント作成が可能です(スキルを公開した場合の連絡先になります)。

入力し終わったら、「保存して続行」をクリックします。

1. プロフィール情報

2. Amazon開発者サービス契約

* は必須項目です。

国・地域 *	日本
名 *	玲央
姓 *	風間
Eメールアドレス *	XXXXXXXXXX.co.jp
電話番号 * e.g. 212-555-1212, +44 0161 715 3369	XXXXXXXXXX
Fax番号	
開発者の氏名または会社名 * アプリやスキルをAmazon.comに公開する際には、この名前が表示されます。	マネージメントサービス株式会社
開発者の氏名または会社名 (ふりがな) * 開発者または会社名のふりがな	まねーじめんとさーびすかぶしきがいしゃ
開発者または会社の説明 最大文字数 4000, 残り: 4000	
住所1 *	丸の内1丁目7番12号
住所2	
都市 *	千代田区
都道府県 *	東京都
郵便番号 *	100-0005
カスタマーサポート用Eメールアドレス	
カスタマーサポート用電話番号	
カスタマーサポート用ウェブサイト	

キャンセル

保存して続行

Agreementを承認

「承認して続行」をクリックします。

The screenshot shows the Amazon Developer Services Agreement page. At the top, there's a navigation bar with the Amazon Developer logo and a search icon. Below the navigation bar, there are two tabs: '1. プロフィール情報' (Profile Information) and '2. Amazon開発者サービス契約' (Amazon Developer Services Agreement). The '2. Amazon開発者サービス契約' tab is selected. Below the tabs, there's a language selector with options for English, 中文 (Chinese), and 日本語 (Japanese). The main content area displays the 'Amazon Developer Services Agreement' text. The text is in English and describes the agreement between Amazon and the developer. At the bottom of the page, there are two buttons: 'キャンセル' (Cancel) and '承認して続行' (Accept and Continue). The '承認して続行' button is highlighted with a red box. Below the buttons, there's a note in Japanese: '日本語は参考訳となり、契約上の束縛力はありません。下の "Accept and Continue" をクリックすることにより、英語版のMobile App Distribution Agreementに同意することとなります。' (Japanese is a reference translation and does not have legal binding force. By clicking "Accept and Continue" below, you agree to the English version of the Mobile App Distribution Agreement.)

開発者プロフィールの 作成完了

「後で行う」をクリックします。



Amazon Developer ダッシュボード

開発者プロフィールの作成が完了すると、Amazon Developer (<https://developer.amazon.com/dashboard>) のダッシュボードに遷移します。

以上でAmazon Developerアカウントの登録は完了です。

amazondeveloper

ダッシュボード アプリ&サービス Alexa Login with Amazon Dash Services レポート 設定

まもなく、支払い情報の更新時に携帯電話のSMSを使った2段階認証が必須となります。携帯電話番号の更新は、[マイアカウント](#)から行えます。[よくある質問](#)（日本語版は下部に記載）もご参照ください。

通知

All	Critical
通知はありません	

アナウンス

Build Dialogs with Less Effort, Less Code, and Less Training Data	2019/06/05	In-Skill Purchasing (ISP) Expands to the UK, German and Japanese Skill Stores	2019/05/30
Alexa Skill Blueprints Now in India	2019/05/07	Alexa Skills Kit Expands to Include Spanish in the US	2019/04/29
Alexa Skills Kit Expands to Brazil	2019/04/10	Alexa Live - Free Online Developer Conference	2019/04/04

Alexa (Amazonの音声サービス、Amazon Echoの頭脳)のための開発をしよう

Alexa Skills Kit
セルフサービスAPI、ツール、資料、コードサンプルを提供しています。Alexaスキルを簡単に開発できます。

Amazonアプリストアで、アプリやゲームを公開しよう

アプリ一覧
すべてのアプリの一覧を見る

AmazonのECサイトからリワードを送れるキャンペーンを実施しよう（現時点では英語のみ）

Momentsのリワード
キャンペーンの準備やリワードの選定について学ぶ

対話モデルの作成

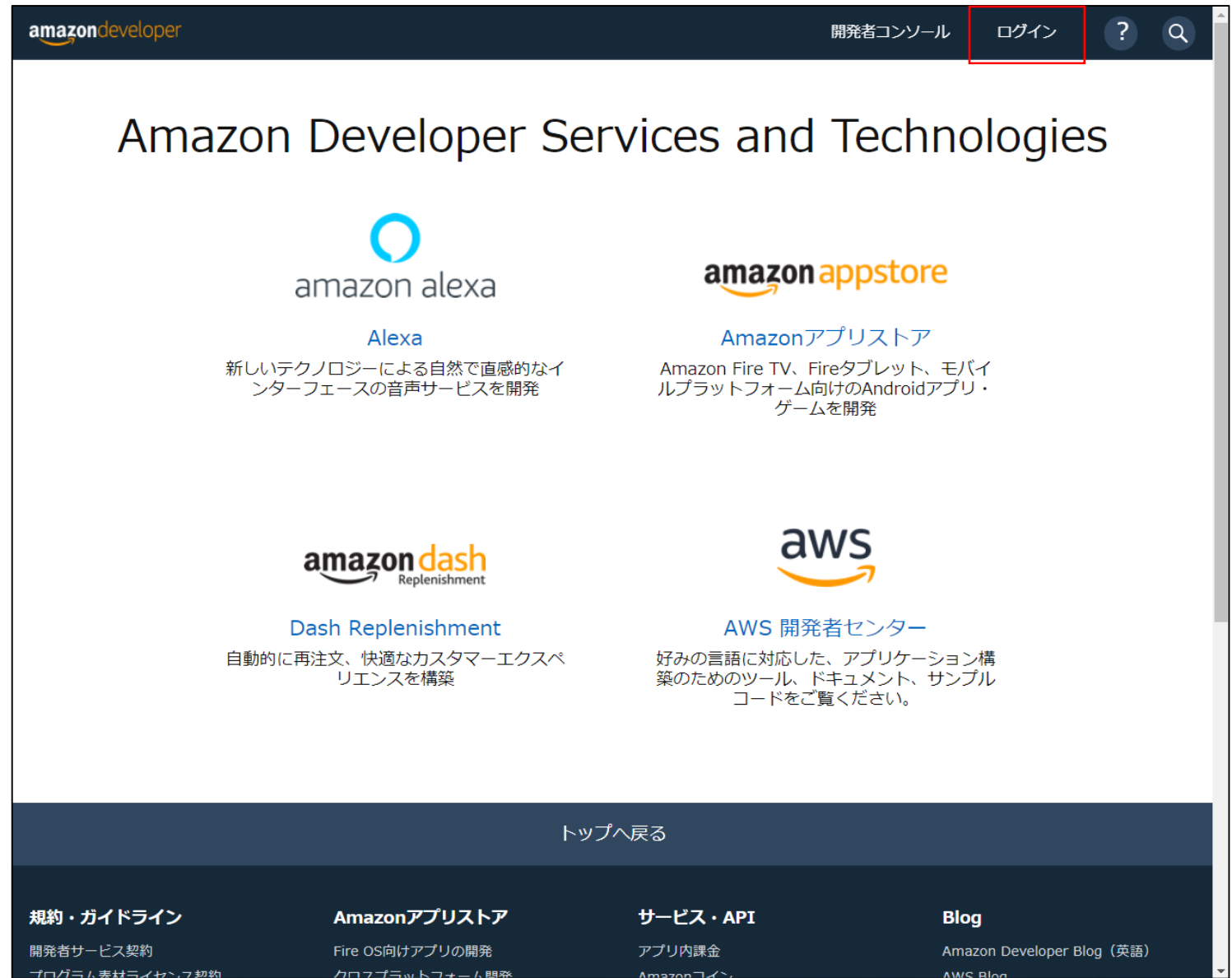
Amazon Developer にアクセス

Amazon Developer にログイン
していない場合、

Amazon Developer
(<https://developer.amazon.com/ja/>)にアクセスします。

(「amazon developer」で検索
すると出てきます。)

「ログイン」をクリックします。



ログイン

開発者プロフィールを作成した
Amazonアカウントでログインし
ます。



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

続行することで、Amazonの[利用規約](#)および[プライバシー規約](#)に同意するものとみなされます。

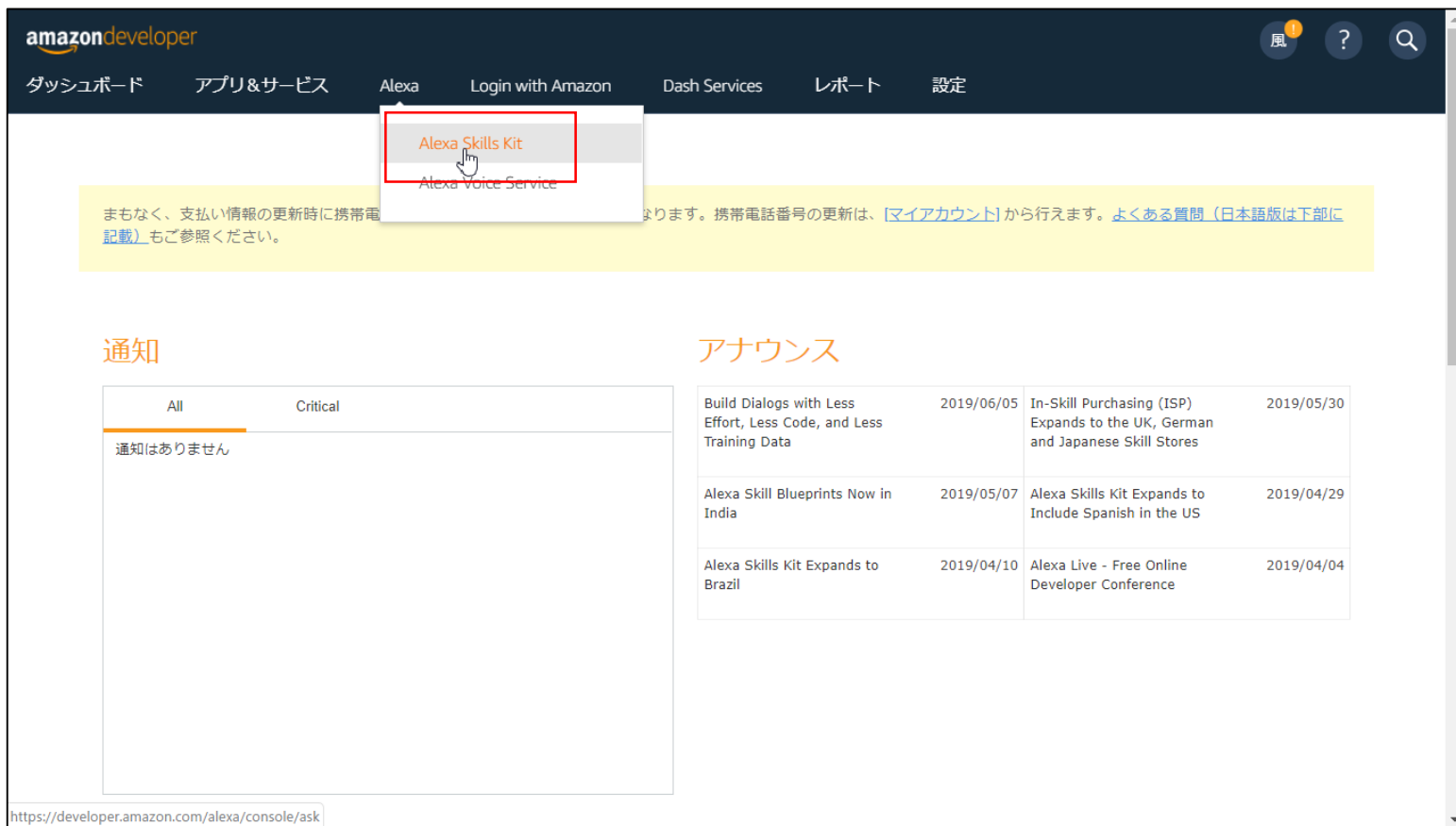
Amazon Developerの新しいお客様ですか？

Amazon Developerアカウントを作成

ダッシュボードから Alexa Skills Kitへ

Amazon Developer にログイン
すると、ダッシュボード
(<https://developer.amazon.com/dashboard>)に遷移します。

Alexaスキルを開発するには、
Alexaタブの「Alexa Skills Kit」をク
リックします。



スキルの作成

alex developer console
(<https://developer.amazon.com/alexa/console/ask>) から

「スキルの作成」をクリックします。



スキル名、言語、モデルを入力・選択

任意のスキル名を入力して「スキルを作成」をクリックします。デフォルトの言語、スキルに追加するモデルはそのまま「日本語(日本)」と「カスタム」にしておきます。

下にスクロールします。

alexa developer console

全画面表示を終了するには **F11** を押します

フィードバックフォーラム

新しいスキルを作成

キャンセル **スキルを作成**

スキル名

エコー 3/50

デフォルトの言語

日本語(日本) ▼

作成後に言語を追加することもできます。

スキルに追加するモデルを選択

自分だけのカスタムモデルを設計したり、プリビルドモデルを使用したりなど、さまざまな方法でスキルの作成を始められます。プリビルドモデルとは、インテントと発話のパッケージを含み、スキルに追加できる対話モデルです。

カスタム 選択済み

ユニークなスキルを作成しましょう。カスタムモデルでスキルの対話をすべて作成できます。

フラッシュブリーフィング

ユーザーが自分のニュースフィードを管理できます。このプリビルドモデルを使用すると、聞きたいコンテンツを自分で管理できるほか、ジャンル別にトピックを検索できます。

"アレクサ、フラッシュニュースを教えてください"

スマート ホーム

ユーザーが自分のスマートホームデバイスを管理できます。このプリビルドモデルを使用すると、照明やその他のデバイスの電源を座ったままオフにできます。

"アレクサ、キッチンライトをオンにして。"

スキルのバックエンドリソースをホスティングする方法を選択

ユーザーのバックエンドリソースをプロビジョニングすることも、Alexaにホストさせることもできます。Alexaにホストさせる場合は、コードエディターを使用できるようになります。コードエディターを使用すると、開発者コンソールから直接AWS Lambdaにコードをデプロイできます。

ホスティングする方法を選択

「Alexaがホスト」を選択したら、上にスクロールして「スキルを作成」をクリックします。

alexa developer console

フィードバックフォーラム

新しいスキルを作成

キャンセル

スキルを作成

ユニークなスキルを作成しましょう。カスタムモデルでスキルの対話をすべて作成できます。

グループ

ユーザーが自分のニュースフィードを管理できます。このプリビルドモデルを使用すると、聞きたいコンテンツを自分で管理できるほか、ジャンル別にトピックを検索できます。

ユーザーが自分のスマートホームデバイスを管理できます。このプリビルドモデルを使用すると、照明やその他のデバイスの電源を座ったままオフにできます。

スキルのバックエンドリソースをホスティングする方法を選択

ユーザーのバックエンドリソースをプロビジョニングすることも、Alexaにホストさせることもできます。Alexaにホストさせる場合は、コードエディターを使用できるようになります。コードエディターを使用すると、開発者コンソールから直接AWS Lambdaにコードをデプロイできます。

ユーザー定義のプロビジョニング

ユーザースキルのエンドポイントとバックエンドリソースをプロビジョニングします。このオプションを使用すると、コンソールのコードエディターを使用できません。

Alexaがホスト

AlexaはAWSの無料上限枠までユーザーのアカウントのスキルをホスティングします。AWS Lambdaエンドポイント、メディアストレージ5GB、毎月のデータ転送量15GB、セッション永続性用のテーブルにアクセスできます。詳しくはこちら

Japanese (日本語) © 2010 - 2019, Amazon.com, Inc. or its affiliates. All Rights Reserved. 無断複写・転載を禁じます 利用規約 ドキュメント フォーラム Alexa開発者ブログ Alexa開発者ホーム

呼び出し名を選択

「呼び出し名」をクリックします。

alex developer console

スキル一覧 エコー ビルド コードエディタ テスト 公開 認定 レポート

フィードバックフォーラム

日本語

カスタム

対話モデル

呼び出し名

インテント(5) 追加

HelloWorldIntent

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(1) 追加

AMAZON.SearchQuery

JSONエディター

インターフェース

エンドポイント

インテント履歴

画面表示 ベータ

スキル内商品

開発を開始するには



リソース

収益を得る

スキル内商品を作成して、スキルに追加すると、音声でユーザーにプレミアムコンテンツを販売できるようになります。

ドキュメント

カスタムスキルのビルドについての詳細は、技術資料を参照してください。

Alexaのサンプルプロジェクト(英語)

GitHubのAlexaプロジェクトを使用すれば、すぐに開始できます。

Alexa Presentation Language (APL)ドキュメント ベータ版

機能をフルに活用した、レスポンスでインタラクティブな画面表示のスキルを作成するには、APL技術資料を使用してください。

Alexaデザインガイドベータ版

APLを使用して、スキルに魅力的でクリエイティブな視覚要素を含む応答をデザインする方法については、こちらをご覧ください。

オンラインサポート(英語)

質問やアイデアをお寄せください。Amazonがお手伝いいたします。

スキルビルダーのチェックリスト

これらのステップを完了すると、シミュレーターのデス
トタブ、またはEchoデバイスを使用して、スキルをデス
トできるようになります。

1.呼び出し名 必須

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 必須

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 必須

正常に対話モデルをビルドします

4.エンドポイント 必須

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

スキル内商品

スキル内商品を作成して、スキルに追加し
ます

呼び出し名を入力

スキルを呼び出すときに話しかける、呼び出し名を入力します。何でもよいですが、「呼び出し名の要件」(画面参照)は満たす必要があります。

入力後、「モデルをビルド」をクリックします。

The screenshot shows the Alexa Developer Console interface. The top navigation bar includes links for 'スキューン', 'エコー', 'ビルド', 'コードエディタ', 'テスト', '公開', '認定', and 'レポート'. The 'ビルド' tab is active. On the left sidebar, the 'カスタム' section is expanded, showing '対話モデル' and '呼び出し名'. Under '呼び出し名', there are lists for 'インテント(5)' and 'ビルトインインテント(4)'. The 'インテント(5)' list includes 'HelloWorldIntent', 'AMAZON.CancelIntent', 'AMAZON.HelpIntent', 'AMAZON.StopIntent', and 'AMAZON.NavigateHomeIntent'. The 'ビルトインインテント(4)' list includes 'AMAZON.NavigateHomeIntent'. The 'スロットタイプ(0)' and 'JSONエディター' sections are also visible. The main content area is titled '呼び出し名' and contains the following text: '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。' Below this text is a sample utterance: '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'. The 'スキルの呼び出し名' field is highlighted with a red box and contains the text 'こだま'. A red arrow points from the 'モデルをビルド' button in the top navigation bar to the '呼び出し名' field. A light blue box at the bottom right contains the '呼び出し名の要件' (Invoke Name Requirements) section, which lists the rules for the invoke name.

alex developer console

スキューン エコー **ビルド** コードエディタ テスト 公開 認定 レポート フィードバックフォーラム

日本語

モデルを保存 パージョン **モデルをビルド** 発話プロファイラー

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 ?

こだま

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。

HelloWorldIntentを確認

ビルドを待っている間に_intent_を見てください。
「HelloWorldIntent」をクリックしてください。

The screenshot shows the Alexa Developer Console interface. The left sidebar contains a navigation menu with options like '日本語', 'カスタム', '対話モデル', '呼び出し名', 'ビルトインintent(4)', 'スロットタイプ(0)', 'JSONエディター', 'インターフェース', 'エンドポイント', 'intent履歴', '画面表示', 'スキル内商品', and 'アカウントリンク'. The '呼び出し名' (Invocation Name) section is selected, showing a list of intents. 'HelloWorldIntent' is highlighted with a red box. The main content area displays the '呼び出し名' (Invocation Name) configuration for 'HelloWorldIntent'. It includes a text input field with the value 'にだま' and a 'スキルの呼び出し名' (Skill Invocation Name) field. A green notification banner at the bottom right states 'クイックビルドに成功しました' (Quick Build successful) and '2019年8月9日金曜日 17:28'.

alex developer console

くスキル一覧 エコー ビルド コードエディタ テスト 公開 認定 レポート フィードバックフォーラム

日本語

モデルを保存 パージョン トレーニングモデル 発話プロファイラー

カスタム

対話モデル

呼び出し名

intent(5) + 追加

HelloWorldIntent

ビルトインintent(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

インターフェース

エンドポイント

intent履歴

画面表示 ペータ

スキル内商品

アカウントリンク

呼び出し名

特定のcustom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 ?

にだま

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。custom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要があります。customスキルの対話モ

クイックビルドに成功しました

クイックビルドが完了しました。発話のサンプルをテストできるようになりました。

2019年8月9日金曜日 17:28

インテントとは

スキルで実行するアクションのことをインテントと言います。

サンプル発話に登録された言葉を話しかけると、インテントが呼ばれます。

デフォルトでは、「Hello」「how are you」など7個のインテントが登録されています。

The screenshot shows the Alexa Developer Console interface for configuring a skill. The left sidebar contains navigation options like '日本語', 'カスタム', '対話モデル', '呼び出し名', 'インテント(5)', 'HelloWorldIntent', 'ビルトインインテント(4)', 'スロットタイプ(0)', 'JSONエディター', 'インターフェース', 'エンドポイント', 'インテント履歴', '画面表示', 'スキル内商品', and 'アカウントリンク'. The main content area is titled 'インテント / HelloWorldIntent' and shows 'サンプル発話(7)'. A list of sample utterances is displayed: 'hello', 'how are you', 'say hi world', 'say hi', and 'hi'. Below this, there is a section for 'ダイアログデリゲートのルール' and 'インテントスロット(0)'. The 'インテントスロット(0)' section includes a table with columns: '順序', '名前', 'スロットタイプ', and 'アクション'.

順序	名前	スロットタイプ	アクション
----	----	---------	-------

テストへ

「完全ビルドが完了しました」と表示されたら、「テスト」をクリックします。

alexa developer console

スキル一覧 エコー ビルド コードエディタ テスト 公開 認定 レポート フィードバックフォーラム

日本語

モデルを保存 バージョン モデルをビルド 発話プロファイラー

Intent / HelloWorldIntent

サンプル発話(7) 一括編集 書き出し

このIntentの呼び出しに使われると考えられる発話

hello	
how are you	
say hi world	
say hi	
hi	

1 - 5 / 7 すべて表示

Dialog Delegateのルール

このIntentでは、Dialog管...

これが無効にされている理由

完全ビルドが完了しました

新しい変更を加えた場合は、モデルを再ビルドして変更を反映させる必要があります。

[発話プロファイラーを使用してモデルと対話](#)

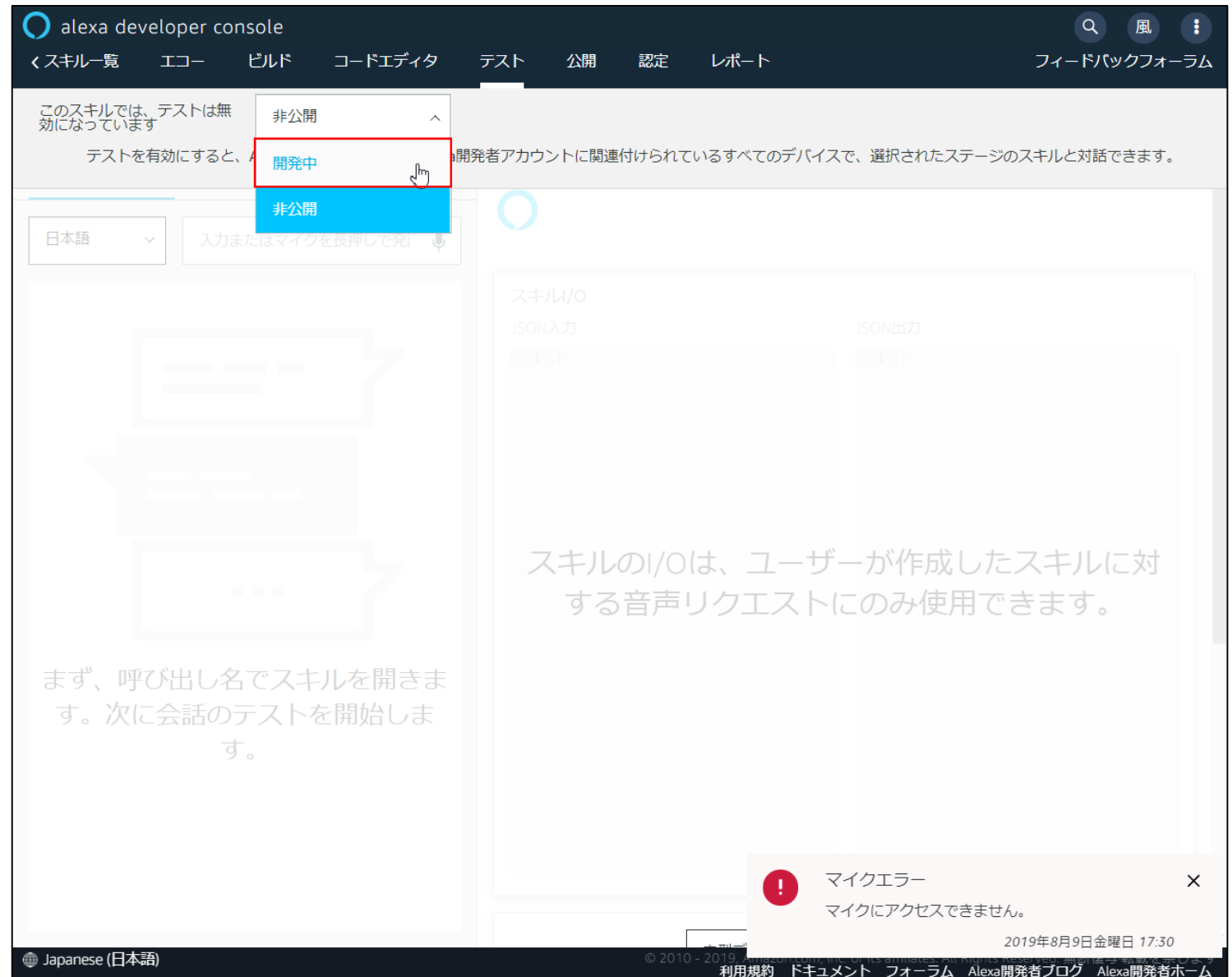
2019年8月9日 17:28

Intentスロット(0)

順序	名前
----	----

テストを有効化

デフォルトではテストが無効になっているので、リストボックスから「開発中」を選んでテストを有効にします。



テストを実行

テキストを入力してEnterでAlexaとのやり取りをシミュレーションできます。

[呼び出し名(ここでは「こだま」)]でスキルを呼び出し、サンプル発話に登録されている発話(ここでは「hello」)でIntentを呼び出します。

やり取りを確認したら、「ビルド」タブに戻ります。

The screenshot shows the Alexa Developer Console interface. The top navigation bar has tabs for <スキル一覧, エコー, **ビルド**, コードエディタ, テスト, 公開, 認定, and レポート. The 'ビルド' tab is highlighted. Below the navigation bar, there's a section for 'スキルテストが有効になっているステージ:' with a dropdown menu set to '開発中'. To the right, there are checkboxes for 'スキル/O' (checked), 'デバイスの表示' (checked), and 'デバイスのログ' (unchecked). The main area is divided into three tabs: 'Alexaシミュレータ', 'JSONエディタ', and '音声と語調'. The 'Alexaシミュレータ' tab is active, showing a chat interface. The input field contains 'Hello World!'. The output shows a response from the skill: 'Welcome, you can say Hello or Help. Which would you like to try?'. The 'JSON出力' section displays the JSON response from the skill, including session information and the output speech: '<say>Hello World!'. The bottom of the console shows a dropdown menu for '中型デバイス'.

intent を追加

「intent」の右にある「+ 追加」をクリックします。

alexa developer console

スキル一覧

エコー

ビルド

コードエディタ

テスト

公開

認定

レポート

フィードバックフォーラム

日本語

カスタム

対話モデル

呼び出し名

intent (5)

+ 追加

HelloWorldIntent

ビルトインintent (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

インターフェース

エンドポイント

intent履歴

画面表示

ベータ

スキル内商品

開発を開始するには

Alexa開発者コンソール...

Intents / PetMatchIntent

Sample Utterances (0)

このintentにはサンプル発話がありません。

リソース

収益を得る

ドキュメント

Alexaのサンプルプロジェクト(英語)

Alexa Presentation Language (APL)ドキュメント

ベータ版

Alexaデザインガイド

ベータ版

オンラインサポート(英語)

スキルビルダーのチェックリスト

これらのステップを完了すると、シミュレーターのデスクトップ、またはEchoデバイスを使用して、スキルをテストできるようになります。

1.呼び出し名

スキルの呼び出し名を入力します

完了

2.intent サンプル

少なくとも1つのintent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

完了

スキル内商品

スキル内商品を作成して、スキルに追加します

developer.amazon.com/alexa/console/ask/build/custom/.../endpoint

カスタム_intentを作成

テキストボックスに「EchoIntent」と入力して「カスタム_intentを作成」をクリックします。

The screenshot shows the Alexa Developer Console interface. The left sidebar contains navigation options: カスタム (Custom), 対話モデル (Dialog Model), インテンツ (Intent), スロットタイプ (Slot Type), JSONエディター (JSON Editor), インターフェース (Interface), エンドポイント (Endpoint), インテンツ履歴 (Intent History), 画面表示 (Screen Display), スキル内商品 (Skill Item), and アカウントリンク (Account Link). The main content area is titled 'intentを追加' (Add Intent) and explains that an intent is an action performed by a user's request. It offers two options: 'カスタム_intentを作成' (Create Custom Intent) and 'Alexaのビルトインライブラリから既存のintentを使用' (Use Existing Intent from Alexa's Built-in Library). The 'Create Custom Intent' option is selected, and a red box highlights the 'EchoIntent' text input field and the 'カスタム_intentを作成' button. A red arrow points from the text input to the button. Below this, there is a search bar for built-in intents and a table listing them. The table has columns for 'ビルトイン名' (Built-in Name) and '説明' (Description). The first entry is '標準' (Standard) with 17 built-in intents, described as '停止、キャンセル、ヘルプなど一般的な操作のintentです。' (It is an intent for general operations such as stop, cancel, help, etc.).

alexa developer console

くスキル一覧 エコー ビルド コードエディタ テスト 公開 認定 レポート フィードバックフォーラム

日本語

モデルを保存 パージョン モデルをビルド 発話プロファイラー

intentを追加

intent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。intentについての[詳細はこちら](#)。

☒ カスタムintentを作成 [?]

EchoIntent カスタムintentを作成

☐ Alexaのビルトインライブラリから既存のintentを使用 [?]

ビルトインを検索

17/17個のビルトイン

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intentです。

インテントスロットを作成

下にスクロールします。

alexa developer console

スキル一覧 エコー ビルド コードエディタ テスト 公開 認定 レポート フィードバックフォーラム

日本語

カスタム

対話モデル

呼び出し名

Intent (6) + 追加

- HelloWorldIntent
- EchoIntent**
- Built-in Intent (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

インターフェース

エンドポイント

Intent履歴

画面表示 ペータ

スキル内商品

Intent / EchoIntent

サンプル発話 (0) 一括編集 書き出し

このIntentの呼び出しに使われると考えられる発話 +

このIntentにはサンプル発話がありません
サンプル発話は、Intent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。

0 - 0 / 0

ダイアログデリゲートのルール

このIntentでは、ダイアログ管... > これが無効にされている理由

Intentスロット (0)

順序	名前	スロットタイプ	アクション
----	----	---------	-------

インテントスロットを追加

インテントスロットの名前に「**phrase**」と入力して「+」ボタンをクリックします。

呼び出し名

▼ インテント(6) + 追加

HelloWorldIntent 🗑️

EchoIntent 🗑️

▼ 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

📱 インターフェース

📍 エンドポイント

📅 インテント履歴

🖼️ 画面表示 🔗 ベータ

スキル内商品

アカウントリンク

アクセス権限

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 +

このインテントにはサンプル発話がありません

サンプル発話は、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。

< 0 - 0 / 0 >

ダイアログデリゲートのルール ?

このインテントでは、ダイアログ管... ▼ > これが無効にされている理由

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	phrase	<div><div></div><div>+</div></div>	<div>スロットタイプを選択 ▼</div> <div>ダイアログを編集 削除</div>

インテントの確認

このインテントには確認が必要です ? 🔘

Japanese (日本語)

© 2010 - 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved. 無断複写・転載を禁じます
利用規約 トクメント フォーラム Alexa開発者ブログ Alexa開発者ホーム

スロットタイプを選択

スロットタイプを「**AMAZON.SearchQuery**」にします。(本来は検索用のスロットタイプです。何でも受け取ることができるので、今回はこちらを利用します。)

このIntentにはサンプル発話がありません

サンプル発話は、Intent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。

0 - 0 / 0

ダイアログデリゲートのルール

このIntentでは、ダイアログ管... > これが無効にされている理由

Intentスロット(1)

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

Intentの確認

このIntentには確認が必要です

Japanese (日本語)

© 2010 - 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved. 無断複写・転載を禁じます
利用規約 トクメント フォーラム Alexa開発者ブログ Alexa開発者ホーム

サンプル発話を追加

サンプル発話に「**{phrase} です**」と入力します。「**{phrase}**」は「**{**」を入力すると出てくる候補から選択できます。その後ろに「**半角スペース+です**」を入力します。

※入力後、「**+**」ボタンをクリックします。

The screenshot shows the Alexa Developer Console interface. On the left sidebar, under 'カスタム' (Custom), the '対話モデル' (Conversation Model) is selected. Under '呼び出し名' (Invocation Name), 'Intent (6)' is expanded, and 'EchoIntent' is selected. Under 'スロットタイプ' (Slot Types), 'AMAZON.SearchQuery' is listed. The main area shows the 'Intent / EchoIntent' configuration. A modal dialog titled '既存のスロットを選択' (Select existing slot) is open, showing a list with 'phrase' selected. A red arrow points from the '+' button in the 'サンプル発話' (Sample Utterances) section to the '+' button in the modal dialog. The 'サンプル発話' section shows 'サンプル発話(0)' and a '+ 一括編集' (Batch Edit) button. Below the modal, there is a text input field for the sample utterance and a '+ 追加' (Add) button. The bottom section shows 'ダイアログデリゲートのルール' (Dialog Delegation Rules) and 'Intent Slots (1)'.

スロットの編集へ

サンプル発話に「**{phrase}** です」が追加されたら、_intentスロットの「**phrase**」をクリックします。

The screenshot shows the Alexa Developer Console interface. The left sidebar contains a navigation menu with options like '日本語', 'カスタム', '対話モデル', '呼び出し名', 'intent(6)', 'EchoIntent', 'phrase', 'ビルトインintent(4)', 'スロットタイプ(1)', 'JSONエディター', 'インターフェース', 'エンドポイント', and 'intent履歴'. The main area displays the 'intent / EchoIntent' configuration. Under 'サンプル発話(1)', a sample utterance '{phrase} です' is shown. A red box highlights this utterance, and a red arrow points from it to the 'intentスロット(1)' table. The table has columns for '順序', '名前', 'スロットタイプ', and 'アクション'. The first row shows '1' in the order column, 'phrase' in the name column (highlighted with a red box), 'AMAZON.SearchQuery' in the slot type column, and 'ダイアログを編集 | 削除' in the action column. The second row shows '2' in the order column, '新しいスロットを作成' in the name column, a '+' sign in the slot type column, and 'ダイアログを編集 | 削除' in the action column. The bottom of the page shows 'intentの確認'.

alexa developer console

スリル一覧 エコー ビルド コードエディタ テスト 公開 認定 レポート フィードバックフォーラム

日本語

モデルを保存 パージョン モデルをビルド 発話プロファイラー

カスタム

対話モデル

呼び出し名

intent(6) + 追加

HelloWorldIntent

EchoIntent

phrase

ビルトインintent(4)

AMAZON.CancelIntent

AMAZON.HelpIntent

AMAZON.StopIntent

AMAZON.NavigateHomeIntent

スロットタイプ(1) + 追加

AMAZON.SearchQuery

JSONエディター

インターフェース

エンドポイント

intent履歴

intent / EchoIntent

サンプル発話(1) 一括編集 書き出し

このintentの呼び出しに使われると考えられる発話 +

{phrase} です

1-1/1

ダイアログデリゲートのルール

このintentでは、ダイアログ管... > これが無効にされている理由

intentスロット(1)

順序	名前	スロットタイプ	アクション
1	phrase	AMAZON.SearchQuery	ダイアログを編集 削除
2	新しいスロットを作成	+	ダイアログを編集 削除

intentの確認

スロットを必須に

「この_intentを完了させるために、このスロットは必須ですか？」をオンにします。

The screenshot shows the Alexa Developer Console interface. The left sidebar contains a navigation menu with options like '日本語', 'カスタム', '対話モデル', '呼び出し名', 'intent(6)', 'EchoIntent', 'phrase', 'ビルトインintent(4)', 'slotタイプ(1)', 'JSONエディター', 'インターフェース', 'エンドポイント', and 'intent履歴'. The 'phrase' slot is selected. The main content area shows the configuration for the 'EchoIntent / phrase' slot. It includes a dropdown for 'スロットのタイプ' set to 'AMAZON.SearchQuery'. A warning message states: 'このintentではオートデリゲートが無効にされています(intent設定に設定)'. Below this, the 'ダイアログ' section shows a message: 'スキルでこれらのプロンプトと発話を使用するには、スキルの応答にDialog.Delegateディレクティブを返します。Dialogディレクティブの使用についての詳細はこちら。'. The 'スロット入力' section has a toggle switch for 'このintentを完了させるために、このスロットは必須ですか？' which is currently turned off. The 'スロットの確認' section has a toggle switch for 'このスロットには確認が必要ですか？' which is also turned off.

alex developer console

スロットを必須に

「このintentを完了させるために、このスロットは必須ですか？」をオンにします。

intent / EchoIntent / phrase

スロットのタイプ

AMAZON.SearchQuery

このintentではオートデリゲートが無効にされています(intent設定に設定)

ダイアログ 検証

スキルでこれらのプロンプトと発話を使用するには、スキルの応答にDialog.Delegateディレクティブを返します。Dialogディレクティブの使用についての詳細はこちら。

スロット入力

このintentを完了させるために、このスロットは必須ですか？

スロットの確認

このスロットには確認が必要ですか？

スロット入力を編集

Alexaの音声プロンプトにAlexaが話しかける任意の言葉を入力して「+」ボタンをクリックします。

さらに、ユーザーの発話に「{phrase}」を入力して「+」ボタンをクリックします。

The screenshot displays the Alexa Skill Builder interface. On the left sidebar, the 'phrase' slot is selected under the 'EchoIntent' category. The main panel shows the 'phrase' slot configuration. At the top, the slot type is set to 'AMAZON.SearchQuery'. A light blue box contains a lightbulb icon and the text: 'このIntentではオートデリゲートがオンになっています(スキル設定から継承)'. Below this, the 'ダイアログ' (Dialog) tab is active, showing a '検証' (Verify) button. The 'スロット入力' (Slot Input) section has a toggle switch for 'このIntentを完了させるために、このスロットは必須ですか?' (Is this slot required to complete this intent?), which is currently turned on. Under 'Alexa の音声プロンプト' (Alexa's voice prompt), there is a text input field with the placeholder 'ユーザーにこのスロットの入力を求めるために、Alexaが話しかける内容を入力してください' and a '+' button. Below this, a sample prompt '何か言ってください' is shown with a trash icon. The 'ユーザーの発話' (User's utterance) section shows a text input field with '{phrase}' and a '+' button, with a red arrow pointing from the '+' button to the 'phrase' slot in the 'スロットの確認' (Verify slot) section. The bottom of the interface shows navigation links for 'スキル内商品', 'アカウントリンク', and 'アクセス権限'.

モデルをビルド

サンプル発話に「**{phrase}** です」が追加されたら、「インターフェース」をクリックします。

The screenshot shows the Alexa Developer Console interface for configuring a custom skill. The left sidebar contains a list of components: カスタム (Custom), 対話モデル (Dialog Model), 呼び出し名 (Invocation Name), インテント(6) (Intents (6)), スロットタイプ(1) (Slot Types (1)), JSONエディター (JSON Editor), インターフェース (Interface), エンドポイント (Endpoint), and インテント履歴 (Intent History). The 'EchoIntent' is selected under Intents. The main content area shows the configuration for 'EchoIntent', including sample utterances, dialog delegates, and intent slots. A sample utterance '{phrase} です' is highlighted with a red box. The 'インターフェース' button in the sidebar is also highlighted with a red box. A red arrow points from the sidebar button to the sample utterance.

alexa developer console

スキル一覧 エコー ビルド コードエディタ テスト 公開 認定 レポート フィードバックフォーラム

日本語

モデルを保存 パージョン モデルをビルド 発話プロファイラー

カスタム

対話モデル

呼び出し名

インテント(6) + 追加

- HelloWorldIntent
- EchoIntent
- phrase
- 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(1) + 追加

- AMAZON.SearchQuery

JSONエディター

インターフェース

エンドポイント

インテント履歴

インテント / EchoIntent

サンプル発話(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考えられる発話 +

{phrase} です

ダイアログデリゲートのルール ?

このインテントでは、ダイアログ管... > これが無効にされている理由

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	phrase	AMAZON.SearchQuery	ダイアログを編集 削除
2	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

インテントの確認

インターフェースへ

画面下部「オートデリゲート」のスイッチボタンをクリックして無効にします。

対話モデル

呼び出し名

▼ インテント(6) + 追加

HelloWorldIntent 🗑

EchoIntent 🗑

phrase 🗑

▼ 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(1) + 追加

AMAZON.SearchQuery 🗑

JSONエディター

インターフェース

🔊 エンドポイント

🔄 インテント履歴

🖼 画面表示 🔗 ベータ

スキル内商品

アカウントリンク

アクセス権限

🇯🇵 Japanese (日本語)

💡 インターフェースを有効にすると、対話モデルに必要なインテントを追加しなければならない場合があります。変更を有効にするには、インターフェースの変更を保存し、モデルを再ビルドするという両方の作業を行う必要があります。

インターフェース名	説明	
Audio Player	AudioPlayerインターフェースでは、オーディオをストリーミングし、再生状況を監視するディレクティブとリクエストを提供します。AudioPlayerインターフェースの詳細はこちら。	<input type="checkbox"/>
Displayインターフェース	画面付きEchoデバイスでは、画面と音声対話の両方を使用するAlexaスキルを作成できます。Displayインターフェースの詳細はこちら。	<input type="checkbox"/>
VideoApp	VideoAppインターフェースでは、Echo ShowのネイティブビデオファイルをストリーミングするVideoApp.Launchディレクティブを提供します。VideoAppインターフェースの詳細はこちら。	<input type="checkbox"/>
Alexa Presentation Language ベータ	Alexa Presentation Language (APL)を使うと、Echo Show、Echo Spot、Fire TV向けのマルチモーダルスキルを開発できます。オーサリングツールとシミュレーターを使って、APLドキュメントを作成、テストしましょう。ベータ版の制約については、 APIリファレンス を参照してください。また、 ブログ (英語) でもこの機能について詳しく説明しています。	<input type="checkbox"/>
オートデリゲート	ダイアログモデルに基づいて、Alexaに自動的にダイアログの各ステップを判断して完成させます。ダイアログが完成すると、単一のIntentRequestを取得します。この設定はインテントレベルで上書きできます。オートデリゲートの詳細はこちら。	<input checked="" type="checkbox"/>

© 2010 - 2019, Amazon.com, Inc. or its affiliates. All Rights Reserved. 無断複写 転載を禁じます

36

モデルをビルド

インターフェイスがすべて無効になったら、「インターフェイスを保存」をクリックします。

その後、「モデルをビルド」をクリックします。

「完全ビルドが完了しました」と表示されたら、対話モデルの作成は完了です(※)。

※「**{phrase}**」と「**です**」の間に半角スペースがなかったり、全角スペースが混ざっていたりすると、エラーになるので注意。

日本語

カスタム

対話モデル

呼び出し名

▼ インテント(6) + 追加

- HelloWorldIntent
- ▼ EchoIntent
- phrase
- ▼ 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(1) + 追加
- AMAZON.SearchQuery
- JSONエディター

インターフェイス

エンドポイント

インテント履歴

画面表示 ベータ

スキル内商品

インターフェイスを保存

バージョン

モデルをビルド

発話プロファイラー

インターフェイス

💡 インターフェイスを有効にすると、対話モデルに必要なインテントを追加しなければならない場合があります。変更を有効にするには、インターフェイスの変更を保存し、モデルを再ビルドするという両方の作業を行う必要があります。

インターフェイス名	説明	
Audio Player	AudioPlayerインターフェイスでは、オーディオをストリーミングし、再生状況を監視するディレクティブとリクエストを提供します。AudioPlayerインターフェイスの詳細はこちら。	<input type="checkbox"/>
Displayインターフェイス	画面付きEchoデバイスでは、画面と音声対話の両方を使用するAlexaスキルを作成できます。Displayインターフェイスの詳細はこちら。	<input type="checkbox"/>
VideoApp	VideoAppインターフェイスでは、Echo ShowのネイティブビデオファイルをストリーミングするVideoApp.Launchディレクティブを提供します。VideoAppインターフェイスの詳細はこちら。	<input type="checkbox"/>
Alexa Presentation Language	Alexa Presentation Language (APL)を使うと、Echo Show、Echo Spot、Fire TV向けのマルチモーダルスキルを開発できます。オーサリングツールとシミュレーターを使って、APLドキュメントを作成、テストしましょう。ベータ版の制約については、 APIリファレンス を参照してください。また、 ブログ (英語) でもこの機能について詳しく説明しています。	<input type="checkbox"/>
オートデリゲート	ダイアログモデルに基づいて、Alexaに自動的にダイアログの各ステップを判断して完成させます。ダイアログが完成すると、単一のIntentRequestを取得します。この設定はインテントレベルで上書きできます。オートデリゲートの詳細はこちら。	<input type="checkbox"/>

AWS Lambda関数(Node.js)の実装

コードエディタへ

「完全ビルドが完了しました」と表示されたら、「コードエディタ」をクリックします。

The screenshot shows the Alexa Developer Console interface. The top navigation bar includes tabs for 'スキル一覧', 'エコ', 'ビルド', 'コードエディタ' (highlighted with a red box), 'テスト', '公開', '認定', and 'レポート'. The 'コードエディタ' tab is selected, and a red arrow points from it to a notification at the bottom right. The notification, titled '完全ビルドが完了しました', contains the text: '新しい変更を加えた場合は、モデルを再ビルドして変更を反映させる必要があります。' and a link '発話プロファイラーを使用してモデルと対話'. The date and time '2019年8月15日 木曜日 9:38' are also shown. The left sidebar shows the 'カスタム' section with '対話モデル' selected, listing various intents and slot types. The main content area displays the 'インターフェース' section with a list of interfaces and their descriptions.

alexa developer console

スキル一覧 エコ ビルド **コードエディタ** テスト 公開 認定 レポート フィードバックフォーラム

日本語

インターフェースを保存 バージョン モデルをビルド 発話プロファイラー

インターフェース

インターフェースを有効にすると、対話モデルに必要な_intent_を追加しなければならない場合があります。変更を有効にするには、インターフェースの変更を保存し、モデルを再ビルドするという両方の作業を行う必要があります。

インターフェース名	説明	
Audio Player	AudioPlayerインターフェースでは、オーディオをストリーミングし、再生状況を監視するディレクティブとリクエストを提供します。AudioPlayerインターフェースの詳細はこちら。	<input type="checkbox"/>
Displayインターフェース	画面付きEchoデバイスでは、画面と音声対話の両方を使用するAlexaスキルを作成できます。Displayインターフェースの詳細はこちら。	<input type="checkbox"/>
VideoApp	VideoAppインターフェースでは、Echo ShowのネイティブビデオファイルをストリーミングするVideoAppLaunchディレクティブを提供します。VideoAppインターフェースの詳細はこちら。	<input type="checkbox"/>
Alexa Presentation Language	Alexa Presentation Language (APL) ドキュメントを使用して、APLドキュメントを開発できます。APLドキュメントの制限について詳しく説明しています。	<input type="checkbox"/>

完全ビルドが完了しました

新しい変更を加えた場合は、モデルを再ビルドして変更を反映させる必要があります。

[発話プロファイラーを使用してモデルと対話](#)

2019年8月15日 木曜日 9:38

Index.jsを編集

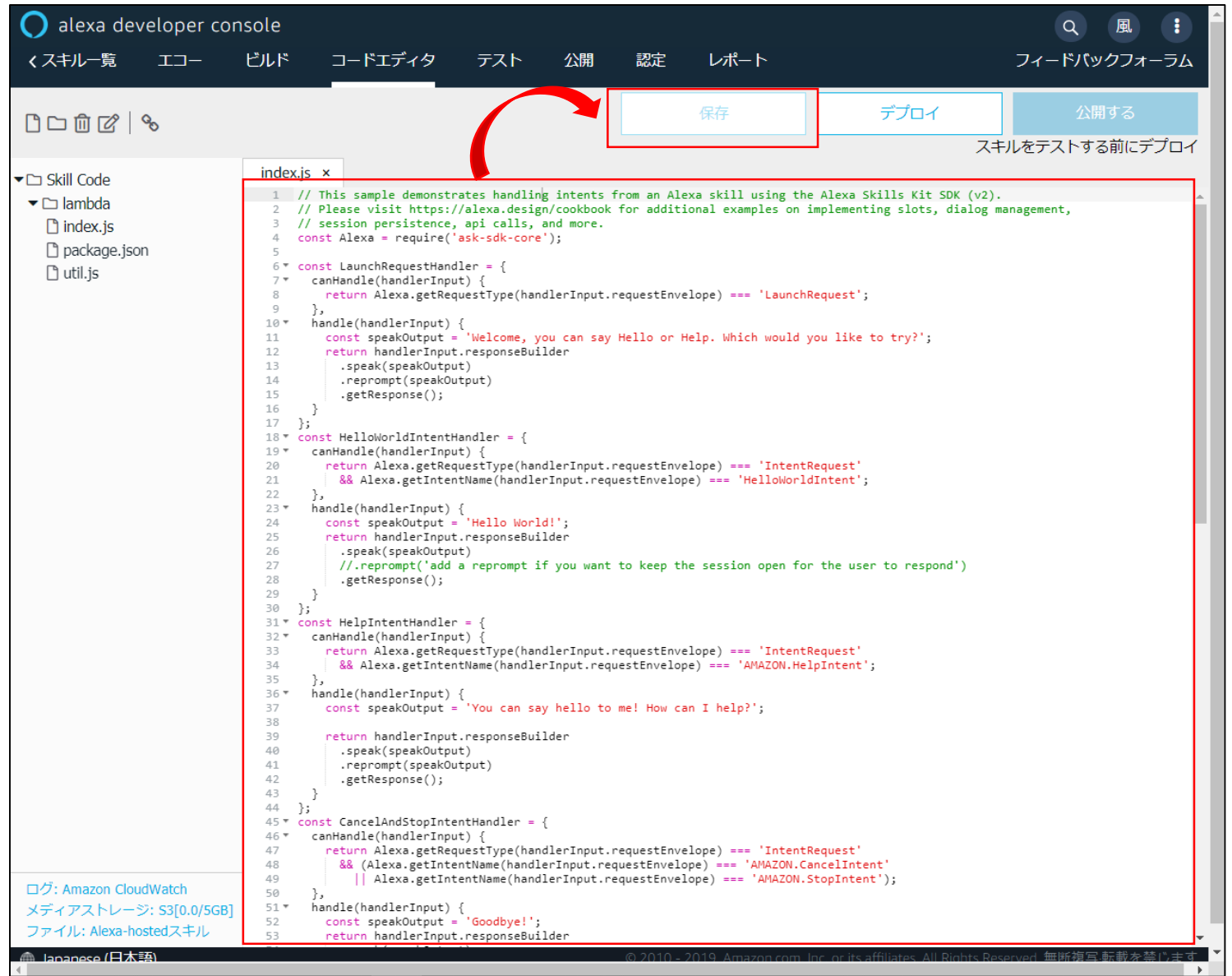
ソースコードの編集画面に
遷移します。

まずは、表示されている
index.jsを編集していきます。

なお、ソースコードは以下の
場所にあります。

<https://github.com/forshoes-admin/Alexa-reception/tree/master/hands-on>

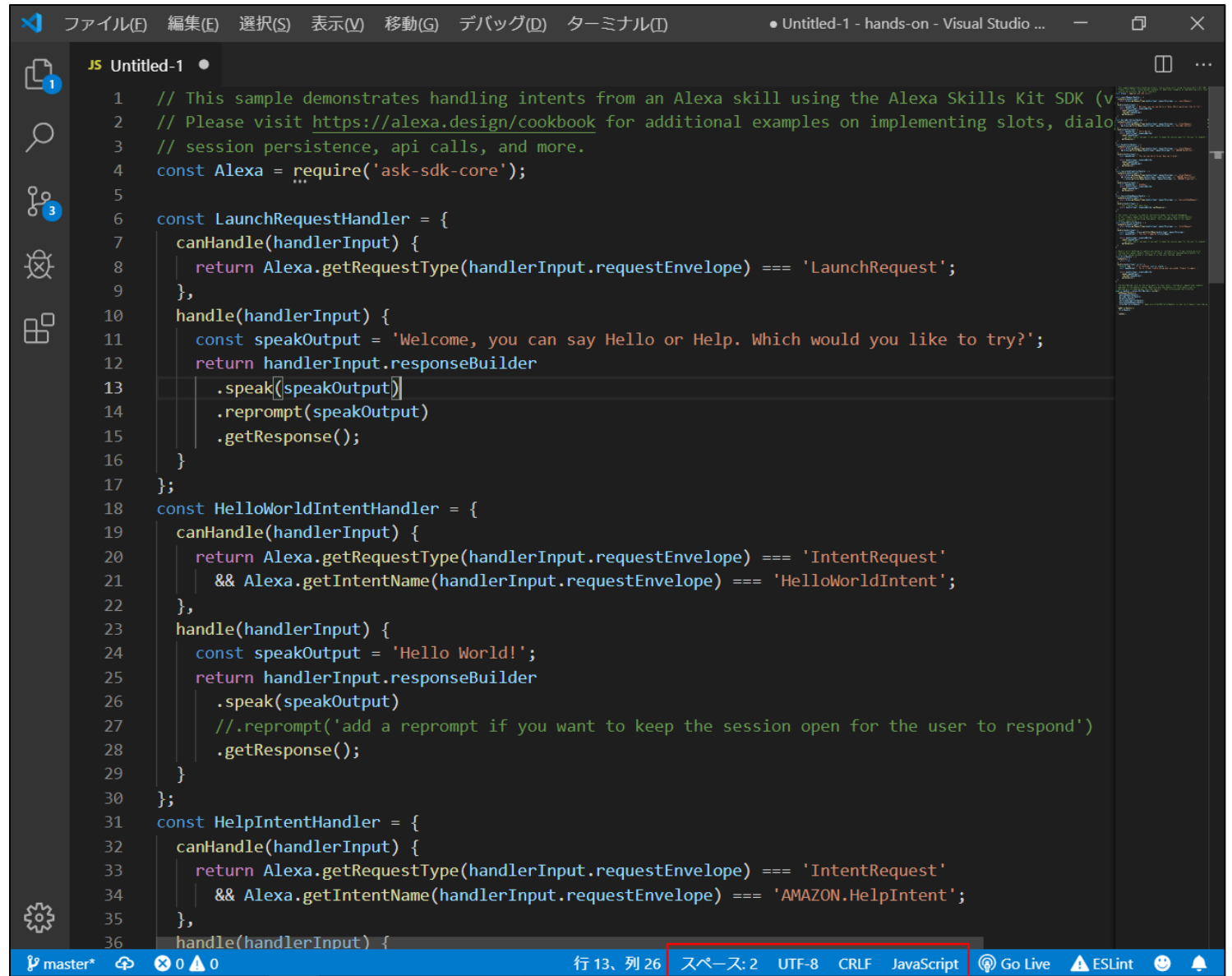
画面上のソースコードを編集したら、「保存」をクリックまたはCtrl + Sで保存できます。



参考: Visual Studio Codeで編集

Index.jsのソースコードを全
選択してコピーし、Visual
Studio Codeに言語を
「JavaScript」、インデントを
「スペース:2」にして張り付け
ると見やすく、編集しやすい
です。

編集が終わったら、Visual
Studio Codeのソースコード
を全選択してコピーし、
alexa developer consoleの
ソースコードを全選択して上
書きで貼り付けます



```
1 // This sample demonstrates handling intents from an Alexa skill using the Alexa Skills Kit SDK (v1.0)
2 // Please visit https://alexa.design/cookbook for additional examples on implementing slots, dialog actions,
3 // session persistence, api calls, and more.
4 const Alexa = require('ask-sdk-core');
5
6 const LaunchRequestHandler = {
7   canHandle(handlerInput) {
8     return Alexa.getRequestType(handlerInput.requestEnvelope) === 'LaunchRequest';
9   },
10  handle(handlerInput) {
11    const speakOutput = 'Welcome, you can say Hello or Help. Which would you like to try?';
12    return handlerInput.responseBuilder
13      .speak(speakOutput)
14      .reprompt(speakOutput)
15      .getResponse();
16  }
17 };
18
19 const HelloWorldIntentHandler = {
20   canHandle(handlerInput) {
21     return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
22       && Alexa.getIntentName(handlerInput.requestEnvelope) === 'HelloWorldIntent';
23   },
24   handle(handlerInput) {
25     const speakOutput = 'Hello World!';
26     return handlerInput.responseBuilder
27       .speak(speakOutput)
28       // .reprompt('add a reprompt if you want to keep the session open for the user to respond')
29       .getResponse();
30   }
31 };
32
33 const HelpIntentHandler = {
34   canHandle(handlerInput) {
35     return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
36       && Alexa.getIntentName(handlerInput.requestEnvelope) === 'AMAZON.HelpIntent';
37   },
38   handle(handlerInput) {
```

master* 0 0 行 13、列 26 スペース: 2 UTF-8 CRLF JavaScript Go Live ESLint

LaunchRequestHandler を編集

LaunchRequestHandler
(`const`
`LaunchRequestHandler = {`
`};`の波括弧の中)を右の通りに編集します。

```
const LaunchRequestHandler = {  
  canHandle(handlerInput) {  
    return Alexa.getRequestType(handlerInput.requestEnvelope) === 'LaunchRequest';  
  },  
  handle(handlerInput) {  
    console.log("LaunchRequestHandlerのhandleを呼び出し");  
    return handlerInput.responseBuilder  
      .addDelegateDirective({  
        name: 'EchoIntent',  
        confirmationStatus: 'NONE',  
        slots: {}  
      })  
      .getResponse();  
  }  
};
```

解説

addDelegateDirective

関数に渡すオブジェクト

`name`プロパティには_intent名を指定します。

`confirmationStatus`プロパティには_intentが確認されたか (CONFIRMED)、拒否されたか (DENIED)を表します。**NONE**の場合は、ユーザーにまだ確認を行っていないことを表します。

`slots`プロパティには、スロット値を指定します。今回はプログラムからスロット値を指定しないので、空のオブジェクトを指定します。

最後に`getResponse`関数を呼び出してレスポンスを返します。

```
const LaunchRequestHandler = {  
.....(省略).....  
  handle(handlerInput) {  
    console.log("LaunchRequestHandlerが起動");  
    return handlerInput.responseBuilder  
      .addDelegateDirective({  
        name: 'EchoIntent',  
        confirmationStatus: 'NONE',  
        slots: {}  
      })  
      .getResponse();  
  }  
};
```

EchoIntentHandler を追加

右の通りの
EchoIntentHandler

(`const EchoIntentHandler = {`
};の波括弧の中)を追加しま
す。

追加する場所は任意ですが、
LaunchRequestHandler

とHelloWorldIntentHandler

の間に追加しました(次ペー
ジ参照)。

```
const EchoIntentHandler = {  
  canHandle(handlerInput) {  
    return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'  
      && Alexa.getIntentName(handlerInput.requestEnvelope) === 'EchoIntent'  
      && Alexa.getDialogState(handlerInput.requestEnvelope) !== 'COMPLETED'  
  },  
  handle(handlerInput) {  
    console.log("EchoIntentHandlerのhandleを呼び出し");  
    let speakOutput = "";  
    try {  
      speakOutput = Alexa.getSlotValue(handlerInput.requestEnvelope, "phrase");  
    } catch (e) {  
      console.error(e);  
    }  
    return handlerInput.responseBuilder  
      .speak(speakOutput)  
      .getResponse();  
  }  
};
```

EchoIntentHandler
を追加する場所

LaunchRequestHandler
とHelloWorldIntentHandler
の間に追加しました。

```
const LaunchRequestHandler = {  
.....(省略).....  
};
```

```
const EchoIntentHandler = {  
  canHandle(handlerInput) {  
.....(省略).....  
  },  
  handle(handlerInput) {  
.....(省略).....  
  }  
};
```

```
const HelloWorldIntentHandler = {  
.....(省略).....  
};
```

解説

`getSlotValue`関数は、`handlerInput.requestEnvelope`とインテントスロット名を引数として呼び出すと、取得したスロットの値が返ってきます。

`speak`関数は、Alexaの応答メッセージを引数として呼び出すと、Alexaが引数の値で応答します。今回は、スロットの値そのままAlexaが応答します。

```
const EchoIntentHandler = {  
    .....(省略).....  
    handle(handlerInput) {  
        console.log("EchoIntentHandlerのhandleを呼び出し");  
        let speakOutput = "";  
        try {  
            speakOutput = Alexa.getSlotValue(handlerInput.requestEnvelope, "phrase");  
        } catch (e) {  
            console.error(e);  
        }  
        return handlerInput.responseBuilder  
            .speak(speakOutput)  
            .getResponse();  
    }  
};
```

index.jsの編集完了

handler関数内の
addRequestHandlers 関数の
引数にEchoIntentHandlerを
追加し、
IntentReflectorHandler(テスト・デバッグ用の関数)をコメントアウトします。

以上でindex.jsの編集は完了です。

```
exports.handler = Alexa.SkillBuilders.custom()
    .addRequestHandlers(
        LaunchRequestHandler,
        EchoIntentHandler,
        HelloWorldIntentHandler,
        HelpIntentHandler,
        CancelAndStopIntentHandler,
        SessionEndedRequestHandler,
        //IntentReflectorHandler, // make sure IntentReflectorHandler is last so it
        //doesn't override your custom intent handlers
    )
    .addErrorHandlers(
        ErrorHandler,
    )
    .lambda();
```

デプロイ

alexa developer consoleで
index.jsの編集が完了したら、
「デプロイ」をクリックしてデ
プロイします。

The screenshot displays the Alexa Developer Console interface. At the top, there's a navigation bar with tabs: < スキル一覧, エコー, ビルド, コードエディタ (selected), テスト, 公開, 認定, レポート. On the right of the navigation bar are icons for search, refresh, and a menu, along with a link to フィードバックフォーラム. Below the navigation bar, there's a toolbar with icons for file operations and three buttons: 保存 (Save), デプロイ (Deploy, highlighted with a red box), and 公開する (Publish). A note on the right says スキルをテストする前にデプロイ (Deploy before testing the skill). The main area is split into two panes. The left pane shows the Skill Code directory structure: Skill Code > lambda > index.js, package.json, and util.js. The right pane shows the code editor for index.js, containing JavaScript code for handling Alexa requests. The code includes comments in Japanese and defines three handlers: LaunchRequestHandler, EchoIntentHandler, and HelloWorldIntentHandler. At the bottom left, there's a status bar showing: ログ: Amazon CloudWatch, メディアストレージ: S3[0.0/5GB], and ファイル: Alexa-hostedスキル. At the bottom right, there's a copyright notice: © 2010 - 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved. 無断複写・転載を禁じます.

```
1 // This sample demonstrates handling intents from an Alexa skill using the Alexa Skills Kit SDK (v2).
2 // Please visit https://alexa.design/cookbook for additional examples on implementing slots, dialog management,
3 // session persistence, api calls, and more.
4 const Alexa = require('ask-sdk-core');
5
6 const LaunchRequestHandler = {
7   canHandle(handlerInput) {
8     return Alexa.getRequestType(handlerInput.requestEnvelope) === 'LaunchRequest';
9   },
10  handle(handlerInput) {
11    console.log("LaunchRequestHandlerが起動");
12    return handlerInput.responseBuilder
13      .addDelegateDirective({
14        name: 'EchoIntent',
15        confirmationStatus: 'NONE',
16        slots: {}
17      })
18      .reprompt()
19      .getResponse();
20  }
21 };
22
23 const EchoIntentHandler = {
24   canHandle(handlerInput) {
25     return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
26       && Alexa.getIntentName(handlerInput.requestEnvelope) === 'EchoIntent'
27       && Alexa.getDialogState(handlerInput.requestEnvelope) !== 'COMPLETED';
28   },
29   handle(handlerInput) {
30     console.log("EchoIntentHandlerのhandleを呼び出し");
31     let speakOutput = "";
32     try {
33       speakOutput = Alexa.getSlotValue(handlerInput.requestEnvelope, "phrase");
34     } catch (e) {
35       console.error(e);
36     }
37     return handlerInput.responseBuilder
38       .speak(speakOutput)
39       .getResponse();
40   }
41 };
42
43 const HelloWorldIntentHandler = {
44   canHandle(handlerInput) {
45     return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
46       && Alexa.getIntentName(handlerInput.requestEnvelope) === 'HelloWorldIntent';
47   },
48   handle(handlerInput) {
49     const speakOutput = 'Hello World!';
50     return handlerInput.responseBuilder
51       .speak(speakOutput)
52       .reprompt('add a reprompt if you want to keep the session open for the user to respond')
53       .getResponse();
54   }
55 };
```

ログ: Amazon CloudWatch
メディアストレージ: S3[0.0/5GB]
ファイル: Alexa-hostedスキル

Japanese (日本語)

© 2010 - 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved. 無断複写・転載を禁じます

デプロイ完了

「デプロイが完了しました」と表示されたら、デプロイ完了です。

早速テストをしてみましょう。

The screenshot shows the Alexa Developer Console interface. At the top, there's a navigation bar with tabs: < スキル一覧, エコー, ビルド, コードエディタ, テスト, 公開, 認定, レポート. The 'テスト' (Test) tab is highlighted with a red box and a red arrow pointing to it. Below the navigation bar, there are buttons: 保存 (Save), デプロイ (Deploy), and 公開する (Publish). A timestamp indicates the final deployment time: 最終デプロイ日時: 8月 19, 2019, 11:57 午前. The main area displays the 'index.js' file content, which is a sample skill implementation. At the bottom right, a green notification box with a checkmark icon states: 'デプロイが完了しました' (Deployment completed). Below this, it says: '新しい変更を加えた場合は、デプロイして変更を反映させる必要があります。' (If you add new changes, you need to deploy to reflect the changes). The notification also includes the date and time: '2019年8月19日月曜日 11:58'.

alexa developer console

< スキル一覧 エコー ビルド コードエディタ **テスト** 公開 認定 レポート

フィードバックフォーラム

保存 デプロイ 公開する

最終デプロイ日時: 8月 19, 2019, 11:57 午前

Skill Code

- lambda
 - index.js
 - package.json
 - util.js

```
index.js x
1 // This sample demonstrates handling intents from an Alexa skill using the Alexa Skills Kit SDK (v2).
2 // Please visit https://alexa.design/cookbook for additional examples on implementing slots, dialog management,
3 // session persistence, api calls, and more.
4 const Alexa = require('ask-sdk-core');
5
6 const LaunchRequestHandler = {
7   canHandle(handlerInput) {
8     return Alexa.getRequestType(handlerInput.requestEnvelope) === 'LaunchRequest';
9   },
10  handle(handlerInput) {
11    console.log("LaunchRequestHandlerが起動");
12    return handlerInput.responseBuilder
13      .addDelegateDirective({
14        name: 'EchoIntent',
15        confirmationStatus: 'NONE',
16        slots: {}
17      })
18      .reprompt()
19      .getResponse();
20  }
21 };
22
23 const EchoIntentHandler = {
24   canHandle(handlerInput) {
25     return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
26       && Alexa.getIntentName(handlerInput.requestEnvelope) === 'EchoIntent'
27       && Alexa.getDialogState(handlerInput.requestEnvelope) !== 'COMPLETED';
28   },
29   handle(handlerInput) {
30     console.log("EchoIntentHandlerのhandleを呼び出し");
31     let speakOutput = "";
32     try {
33       speakOutput = Alexa.getSlotValue(handlerInput.requestEnvelope, "phrase");
34     } catch (e) {
35       console.error(e);
36     }
37     return handlerInput.responseBuilder
38       .speak(speakOutput)
39       .getResponse();
40   }
41 };
42
43 const HelloWorldIntentHandler = {
44   canHandle(handlerInput) {
45     return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
46       && Alexa.getIntentName(handlerInput.requestEnvelope) === 'HelloWorldIntent';
47   },
48   handle(handlerInput) {
49     const speakOutput = 'Hello World!';
50     return handlerInput.responseBuilder
51       .speak(speakOutput)
52       .reprompt('add a reprompt if you want to keep the session open')
53       .getResponse();
54   }
55 };
```

ログ: Amazon CloudWatch
メディアストレージ: S3[0.0/5GB]
ファイル: Alexa-hostedスキル

日本語 (日本語)

© 2010 - 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved. 無断転載を禁じます

デプロイが完了しました

新しい変更を加えた場合は、デプロイして変更を反映させる必要があります。

2019年8月19日月曜日 11:58

「こだま」で呼び出し、「〇〇
です」に対して、Alexaが「〇
〇」と返すのを確認できます。

開発者アカウントでログインしてAlexaアプリを使えば、シミュレーションではなく、実際のAlexaの動作を確認できます。



最後に

Alexaスキル開発についてもっと学びたい場合

- ・今回は「〇〇です」で発話していますが、以下のサイトの方法を使うとテキストをそのまま発話してオウム返しできます。

Alexaで自由テキストをつかむ方法 — おうむ返しくんの場合 —

(<https://note.mu/torusnote/n/n77896a394177>)

以下、公式の学習用資料です。

- ・「サンプル星占い」スキルの作成手順です。

Alexaスキル開発トレーニングシリーズ 第2回 対話モデルとAlexa SDK

(<https://developer.amazon.com/ja/blogs/alexa/post/07377568-2815-4028-8c21-409dd8e84fa2/alexa-training-jp-2nd>)

- ・上記トレーニングシリーズの最新版

Alexaスキル開発トレーニング

(<https://developer.amazon.com/ja/alexa-skills-kit/training/building-a-skill>)

参考

- Amazon Alexaで音声サービスを開発しよう
(<https://developer.amazon.com/ja/alexa-skills-kit>)
- 【Alexa】Connpass用の受付スキルを作ってみた #Alexa #AlexaDevs
(<https://dev.classmethod.jp/voice-assistant/try-to-develop-registration-skill/>)
- Alexaスキル開発トレーニングシリーズ 第2回 対話モデルとAlexa SDK
(<https://developer.amazon.com/ja/blogs/alexa/post/07377568-2815-4028-8c21-409dd8e84fa2/alexa-training-jp-2nd>)
- インテントチェーンを使って別のインテントのダイアログを管理する Dialog.Delegateを使ったインテントのチェーン
(<https://developer.amazon.com/ja/blogs/alexa/post/b3939b11-5cef-4598-b2f9-c7ad904e1692/understanding-intent-chaining>)
- [日本語Alexa] Alexa SDK for Node.js Ver2入門(その3)レスポンスの作成
(<https://dev.classmethod.jp/cloud/alexa-sdk-v2-third/>)
- 【Alexa】ASK SDKにスキルエンジニアの強い味方「ASK SDK Utilities」が登場。 #Alexa #AlexaDevs
(<https://dev.classmethod.jp/voice-assistant/introduce-to-ask-sdk-utilities/>)