
ハンズオン Step2:

TOP ページを APL で構築する

目次

目次	2
【Step2】 TOP ページを APL で構築する	4
Step2 のゴール	4
Step1 で作る対話モデル	4
インターフェースの改修	5
Lambda の改修	7
テスト	11
クイズ	11

【Step2】 TOP ページを APL で構築する

Step2 のゴール

Step2 ではスキルを呼び出した際の TOP ページを APL で構築します。講義の際に作成した APL ドキュメントをスキルから表示させます。

以下のことができるようになります。

- 静的な APL ページをスキルから呼び出せるようになる

Step1 で作る対話モデル

※Step1 と同じです

ポイント

- APL ドキュメントを Lambda 内に JSON ファイルで保持し、requires で呼び出します。

インターフェースの改修

1. 上部メニューより「ビルド」をクリックし、左ペインにある「インターフェース」をクリックします。



2. 画面下部にある「Alexa Presentation Language」をオンにします。終わりましたら画面上部にある「インターフェースを保存」をクリックし「モデルをビルド」をクリックします。

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。	
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。	
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。	
Alexa Presentation Language	<div>ベータ</div> 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、 APL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。	<div>①</div> 

②

③

インターフェースを保存

モデルをビルド

発話プロファイラー

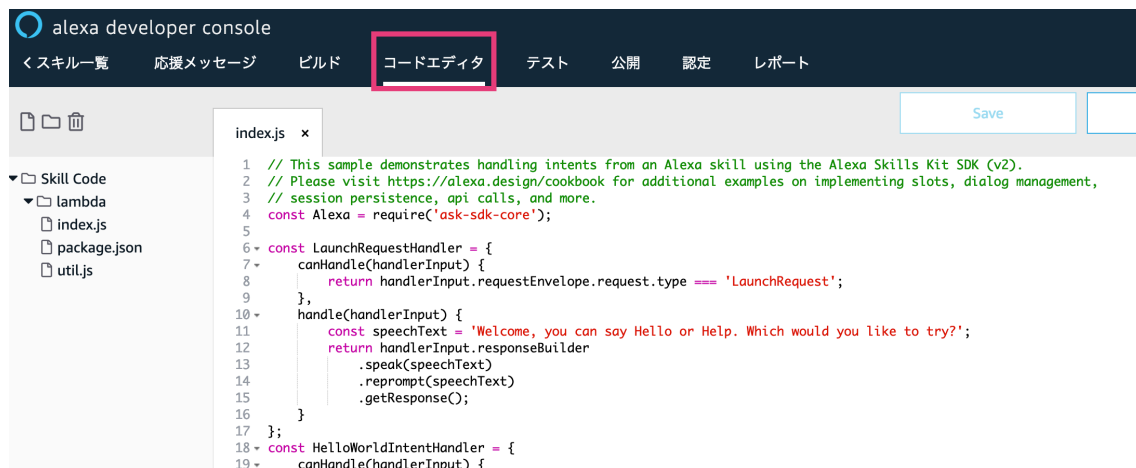
インターフェース


 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

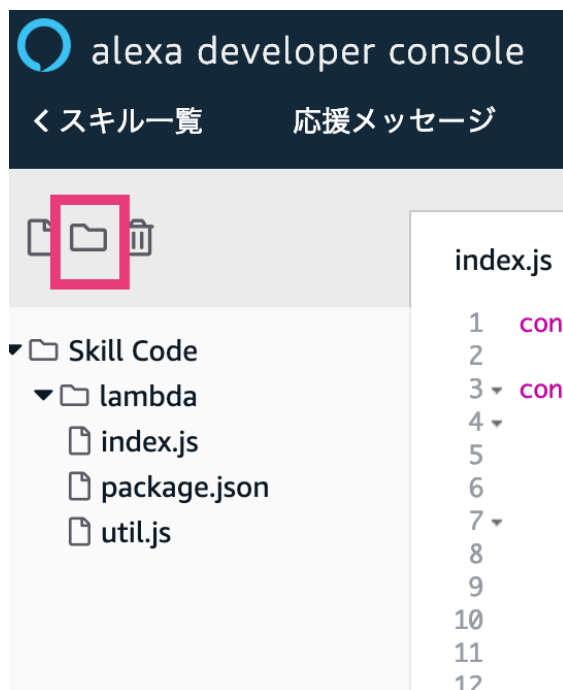
これでインターフェースの設定は完了です。次に Lambda を改修します。

Lambda の改修

1. 上部メニューより「コードエディタ」をクリックし、Lambda のコード画面を表示します。

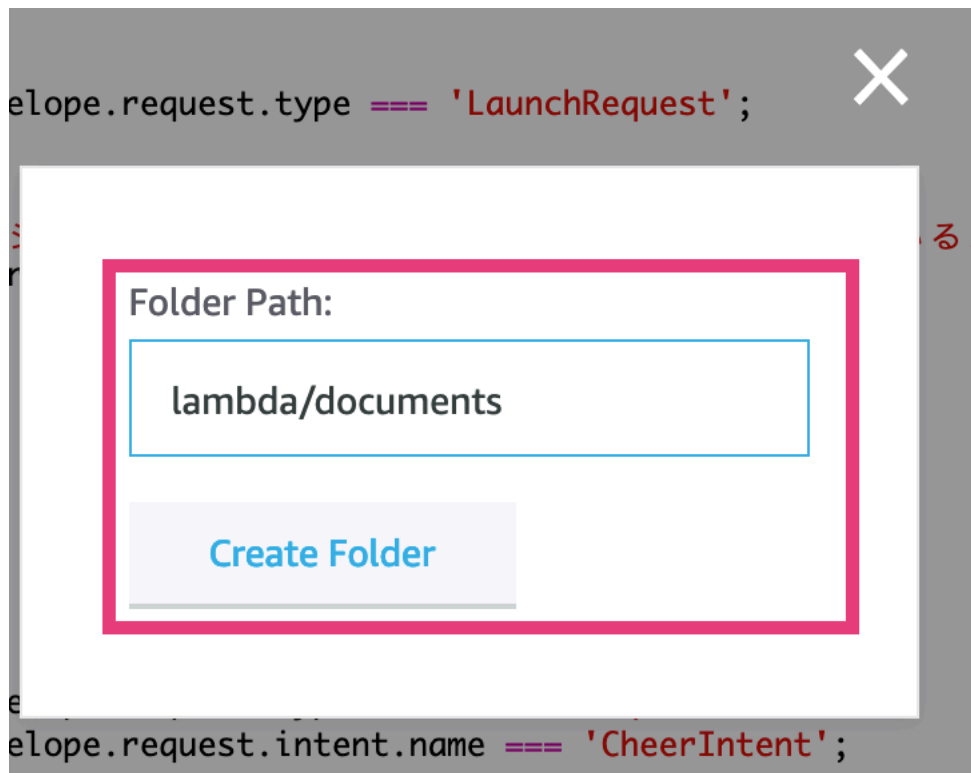


2. Lambda に新規フォルダを作成します。画面左上部にあるフォルダアイコンをクリックします。

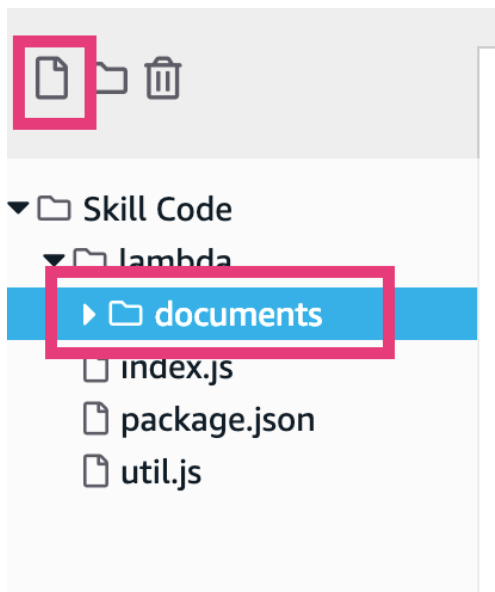


3. ポップアップ画面の「folder path」に「lambda/documents」と入力し「create folder」をクリックします。

※つづり、大文字小文字に注意！

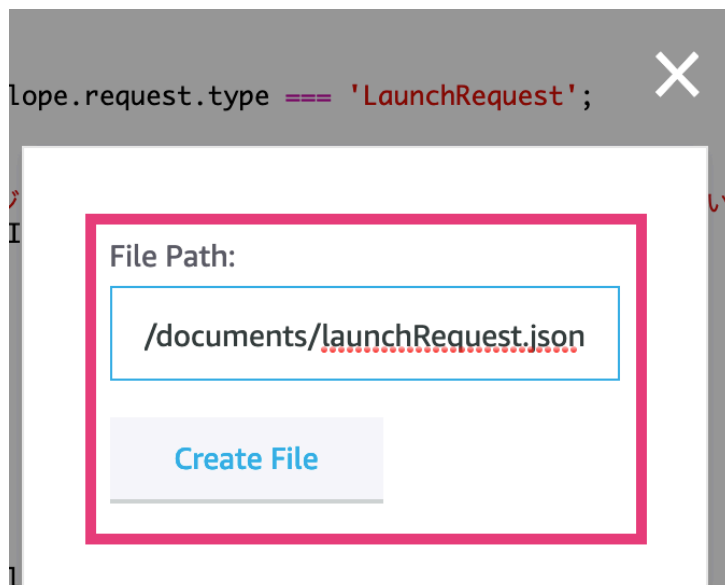


4. 作成した documents フォルダをクリックし、ファイルアイコンをクリックします。



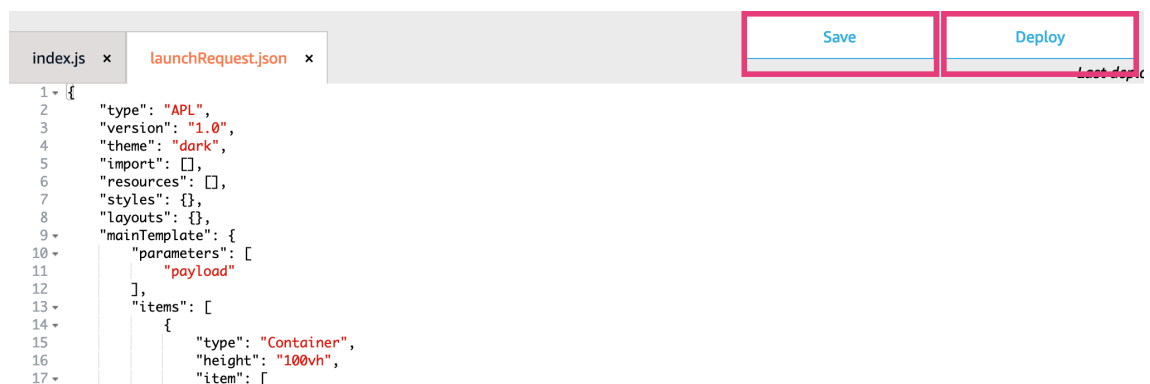
5. 新しく作成した documents フォルダ内に新規 json ファイルを作成します。ポップアップ画面の「file path」に「lambda/documents/launchRequest.json」と入力し「create file」をクリックします。

※つづり、大文字小文字に注意！



6. 出来上がった launchRequest.json にダウンロードしたハンズオン資料より STEP2 -> lambda -> launchRequest.json のファイルを開いてコピー&ペーストします。終わりましたら画面右上部にある「Save」をクリックします。「Deploy」は今どちらでも構わないです。

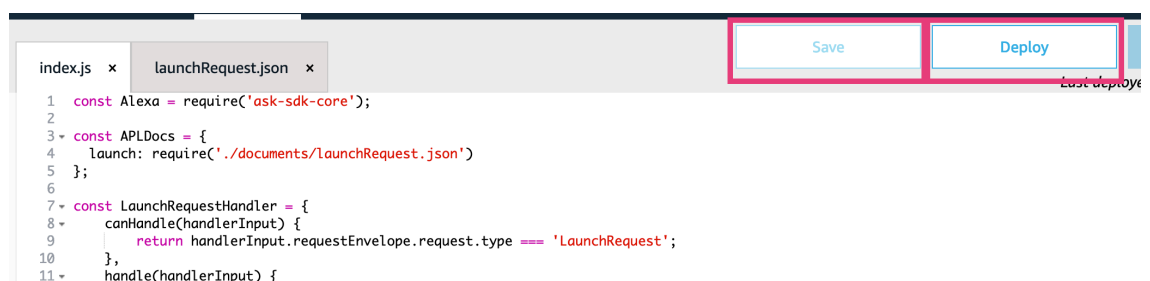
※日本語が文字化けしている方は launchRequest-shift-jis.json を開くか、エディターの文字コードを「UTF-8」にして launchRequest.json を開いてみてください。



```
1 {
2   "type": "APL",
3   "version": "1.0",
4   "theme": "dark",
5   "import": [],
6   "resources": [],
7   "styles": {},
8   "layouts": {},
9   "mainTemplate": {
10    "parameters": [
11      "payload"
12    ],
13    "items": [
14      {
15        "type": "Container",
16        "height": "100vh",
17        "item": [
```

7. index.js を画面上に開き、ダウンロードしたハンズオン資料より STEP2 -> lambda -> index.js のファイルを開いてコピー&ペーストします。

終わりましたら画面右上部にある「Save」をクリックし、その横にある「Deploy」をクリックします。



```
1 const Alexa = require('ask-sdk-core');
2
3 const APLODocs = {
4   launch: require('./documents/launchRequest.json')
5 };
6
7 const LaunchRequestHandler = {
8   canHandle(handlerInput) {
9     return handlerInput.requestEnvelope.request.type === 'LaunchRequest';
10   },
11   handle(handlerInput) {
```

テスト

1. STEP1 同様テストシミュレーターから「応援メッセージ」でスキルを開いて、動いているかどうか確認しましょう。APL で記述した画面が表示されていれば成功です。

※画面のシミュレーターはテストシミュレーターの下の方にあります。画面シミュレーターが見えるところまでスクロールしてください。



クイズ

STEP1 と STEP2 の index.js は具体的にどこが違うでしょう？該当する箇所にコメントを入れてみてください。