## ハンズオン Step4:

# TOPページを APL で動かしてみる



## 目次

目次	2
【Step4】TOPページを APL で動かしてみる	4
Step4 のゴール	4
Step4 で作る対話モデル	4
APL Document の改修	5
Lambda の改修	7
テスト	8
クイズ	8





# 【Step4】TOPページを APL で動か してみる

## Step4 のゴール

Step4 では TOP ページを一定時間ごとにスライドショー形式で動かしてみます。

以下のことができるようになります。

● APL ページによるスライドショーの仕組みを理解する

## Step4 で作る対話モデル

※Step1と同じです

#### ๗ ポイント

• APL Command を使用して自動的にページング処理を行います。

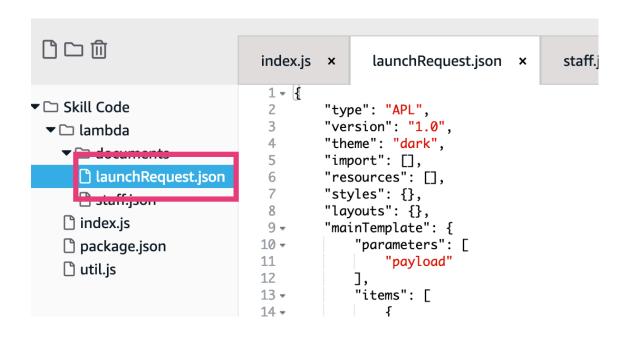


#### APL Document の改修

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。

```
alexa developer console
                                 応援メッセージ
                                                                                                                                                             認定
  くスキル一覧
                                                                   ビルド
                                                                                       コードエディタ
                                                                                                                         テスト
                                                                                                                                             公開
                                                                                                                                                                              レポート
 index.js ×
                                                           // This sample demonstrates handling intents from an Alexa skill using the Alexa Skills Kit SDK (v2).
// Please visit https://alexa.design/cookbook for additional examples on implementing slots, dialog management,
// session persistence, api calls, and more.
const Alexa = require('ask-sdk-core');
▼ 🗀 Skill Code
      index.js
       nackage.json
                                                       6 - const LaunchRequestHandler = {
                                                                  St LaunchRequestrander - \
canHandle(handlerInput) {
    return handlerInput.requestEnvelope.request.type === 'LaunchRequest';
}
       🖺 util.js
                                                                  },
handle(handlerInput) {
    const speechText = 'Welcome, you can say Hello or Help. Which would you like to try?';
    return handlerInput.responseBuilder
        .speak(speechText)
        .reprompt(speechText)
        .getResponse();
}
                                                   10 •
11
12
13
14
15
16
17
18 •
                                                            const HelloWorldIntentHandler = {
    canHandle(handlerInput) {
```

2. documents フォルダの中の launchRequest.json ファイルをダブルクリックして画面上に開きます。





4. 画面上の launchRequest.json にダウンロードしたハンズオン資料より STEP4 -> 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。

※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。

```
Save
                 launchRequest.json ×
index.js ×
                                               staff.json ×
                                                                  package.json ×
                                            "paddingRight": "50px",
"paddingBottom": "50px",
138
140
                                            "textAlign": "right",
141
                                            "text": "しょう"
142
                                       }
143
                                  ]
144
                             },
145 -
                                   "type": "Container",
"id": "seino",
146
147
                                   "justifyContent": "end",
148
                                   "item": [
149 -
150 -
```

これで APL Document の改修は完了です。次は Lambda を改修します。



## Lambda の改修

1. 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 STEP3 -> 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、その横の「Deploy」をクリックします。



## 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。メッセージの途中に画面が変わったら成功です。



## クイズ

STEP3 と STEP4 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。

