ハンズオン Step3:

選択結果ページを APL で構築する



目次

目次	2
【Step3】選択結果ページを APL で構築する	4
Step3 のゴール	4
Step3 で作る対話モデル	4
APL Document の追加	5
Lambda の改修	8
問題	9
ヒント	10
テスト	
クイズ	





【Step3】選択結果ページを APL で 構築する

Step3 のゴール

Step3 では、応援するスタッフを決めた際の選択後のページを APL で構築します。選択結果のページの構築は一枚で、内容を選択結果によって出し分けます。

以下のことができるようになります。

● 動的な APL ページをスキルから呼び出せるようになる

Step3 **で作る対話モデル**

※Step1と同じです

ポイント

• APL ドキュメントを変数化し APL データを Lambda から動的にバインディングさせます。

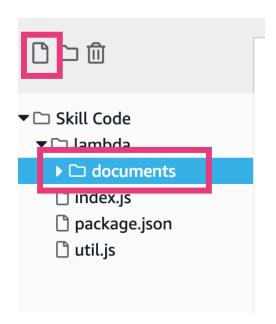


APL Document の追加

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
                                                                           コードエディタ
  くスキル一覧
                            応援メッセージ
                                                                                                         テスト
                                                                                                                          公開
                                                                                                                                        認定
                                                                                                                                                       レポート
 // 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
   ▼ 🗀 lambda
     🗋 index.js
                                               nackage.json
     🖺 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 •
                                                  };
r const HelloWorldIntentHandler = {
    canHandle(handlerInput) {
```

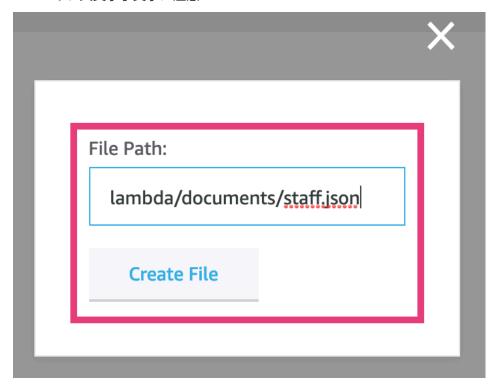
2. documents フォルダをクリックし、ファイルアイコンをクリックします。





3. ポップアップ画面の「file path」に「lambda/documents/staff.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!





- 4. 出来上がった staff.json にダウンロードしたハンズオン資料より STEP3 -> lambda -> staff.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 staff-shift-jis.json を開くか、エディターの文字コードを「UTF-8」にして staff.json を開いてみてください。

これで APL Document の追加は完了です。次は Lambda を改修します。



Lambda の改修

1. package.json をダブルクリックし、package.json を画面上に開きます。

```
index.js ×
                                                   launchRequest.json ×
                                                                                staff.json ×
                                                                                                   package.json

▼ □ Skill Code

                                          "name": "hello-world",
                                          "version": "0.9.0",

▼ □ lambda
                                          "description": "alexa utility for quickly building skills",
    ▶ □ documents
                                          "main": "index.js",
                                   5
                                          "scripts": {
    🖺 indav is
                                            "test": "echo \"Error: no test specified\" && exit 1"
    🗋 package.json
                                   8
                                         "author": "Amazon Alexa",
"license": "ISC",
"dependencies": {
    🔲 utıl.js
                                   9
                                  10
                                  11 -
                                            "ask-sdk-core": "^2.0.7",
"ask-sdk-model": "^1.4.1",
                                  12
                                  13
                                  14
                                           "aws-sdk": "^2.326.0"
                                  15
                                  16 }
```

2. ダウンロードしたハンズオン資料より STEP3 -> lambda -> package.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。

```
ndex.js x launchRequest.json x staff.json x package.json x

1 * {
2     "name": "hello-world",
3     "version": "0.9.0",
4     "description": "alexa utility for quickly building skills",
5     "main": "index.js",
6     "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8     },
9     "author": "Amazon Alexa",
10     "license": "ISC",
1     "dependencies": {
1          "ask-sdk-core": "^2.0.7",
3          "ask-sdk-model": "^1.4.1",
14          "aws-sdk": "^2.326.0",
15          "request": "^2.81.0"

16     }
17     }
```



3. 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 -> question.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] をクリックします。

```
index.js x launchRequest.json x staff.json x

1 const Alexa = require('ask-sdk-core');
2
3 * const APLDocs = {
4     launch: require('./documents/launchRequest.json'),
5     staff: require('./documents/staff.json')
6  };
7
8 * const staffImage = {
9     onsen: "https://s3-ap-northeast-1.amazonaws.com/aajug-apl-handson/onsenbba.jpg",
10     show: "https://s3-ap-northeast-1.amazonaws.com/aajug-apl-handson/show.jpg",
11     seino: "https://s3-ap-northeast-1.amazonaws.com/aajug-apl-handson/seino.jpg"
12  };
13
14 * const staffString = {
15     onsen: "%\theta.",
16     show: "\theta.",
16     show: "\theta.",
17     show: "\theta.",
18     show: "\theta.",
19     show: "\theta.",
10     show: "\theta.",
11     show: "\theta.",
12     show: "\theta.",
13     show: "\theta.",
14     show: "\theta.",
15     show: "\theta.",
16     show: "\theta.",
17     show: "\theta.",
18     show: "\theta.",
19     show: "\theta.",
10     show: "\theta.",
11     show: "\theta.",
12     show: "\theta.",
13     show: "\theta.",
14     show: "\theta.",
15     show: "\theta.",
16     show: "\theta.",
17     show: "\theta.",
18     show: "\theta.",
19     show: "\theta.",
10     show: "\theta.",
11     show: "\theta.",
12     show: "\theta.",
13     show: "\theta.",
14     show: "\theta.",
15     show: "\theta.",
16     show: "\theta.",
17     show: "\theta.",
18     show: "\theta.",
19     show:
```

問題

この question.index.js は「YourNameIntentHandler」の handler ブロックの一部が抜けています。

ここには選択したスタッフの画像と名前を APL ドキュメントにバインディングさせてレスポンスするコードが入ります。

さて、ここにはどのようなコードが入るでしょうか。考えて書き込んでみてください。



ヒント

- 選択したスタッフの画像は変数[img]に、名前は変数[str]に入っています。このデータを先程作成した APL Documnent [staff.json]にバインディングします。
- APL Document を出力するのは「responseBuilder.addDirective()」関数です。
 LaunchRequestHandlerの handler ブロックを参考にしてみてください。
- staff.json にはスタッフの画像を["\${payload.cheerData.properties.staffImage}"(23 行目)]、
 スタッフの名前を["\${payload.cheerData.properties.staffString}"(32 行目)]と定義しています。
- ・ APL Document にデータをバインディングするには「responseBuilder.addDirective()」の ['Alexa.Presentation.APL.RenderDocument'] type 内の[datasources]セクションにデータを 定義するとバインディングされます。

ギブアップ!!という方 -> ダウンロードしたハンズオン資料より STEP3 -> lambda -> answer.js のファイルに正解のコードが載っています。(でもギリギリまでがんばってください!)

それでも動かない!!という方 -> ダウンロードしたハンズオン資料より STEP3 -> lambda -> index.js のファイルを画面上の index.js にコピー&ペーストします

いずれの方もコードの改修が終わったら画面右上部にある「Save」をクリックし、その横の「Deploy」をクリックしてください。



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。選択したスタッフによって最後の画面が変わっていたら成功です。



クイズ

STEP2 と STEP3 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。

