# 日日是Oracle APEX

Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2022年3月23日水曜日

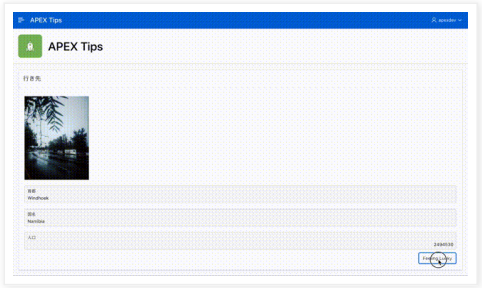
## Developer Tips from the APEX Team #2で紹介されたAJAXを使ったアプリを実装してみる

先日（2022年3月17日）のAPEX Office Hour - Developer Tips from the APEX Team Part IIでVincent Morneauさんが紹介していたAJAXを使ったアプリを、Autonomous DatabaseのAPEX 21.1で作ってみました。

その作業の紹介です。

元のアプリはAPEX 21.2から導入された記法（主にJavaScript API）を使っていて、そのままではAPEX 21.1に実装できません。あと、私はJavaScriptは苦手なので、勉強のために写経をします。

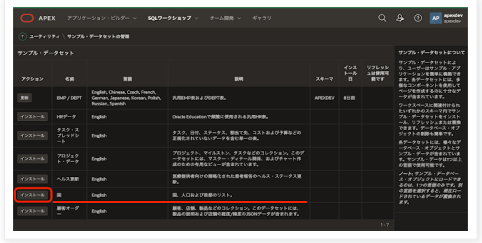
色々と見た目を調整すると手順を書ききれなくなるため、見た目の調整は最低限にします。作成するアプリは以下になります。




最初に**サンプル・データセット**から**国**をロードします。

**SQLワークショップ**の**ユーティリティ**より**サンプル・データセット**を開きます。

**名前**が**国**のデータセットを**インストール**します。



データセットの説明が表示されます。**言語**は**English**しかなく、**スキーマ**もデフォルトのままとします。

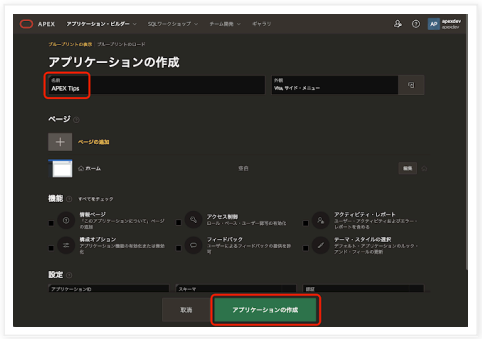
**次**へ進みます。


**データセットのインストール**を実行します。今回のアプリが使用するのはビュー**EBA_COUNTRIES_V**です。


データセットのインストール結果を確認し、**終了**をクリックします。ここではアプリケーションの作成は行いません。

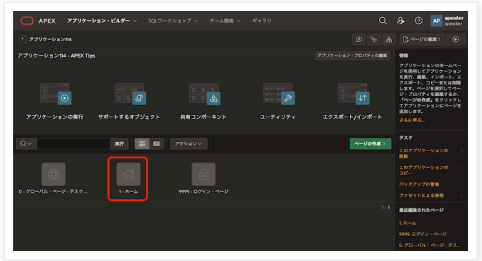

データセットの準備ができたら、アプリケーションを作成します。

**アプリケーション作成ウィザード**を起動し、アプリケーションの**名前**を**APEX Tips**とします。それ以外は何も設定せず、**アプリケーションの作成**を実行します。




アプリケーションが作成されます。今回のアプリケーションは、すべてホーム・ページに実装します。

**ページ・デザイナ**にて**ホーム・ページ**を開きます。

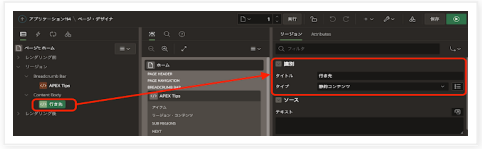


イメージを表示するページ・アイテムと首都、国名、人口を表示するためのページ・アイテムを作成します。

最初にそれらを配置するリージョン**行き先**を作成します。

**Content Body**の上で**コンテキスト・メニュー**を表示させ、**リージョンの作成**を実行します。

作成したリージョンの**識別**の**タイトル**を**行き先**、**タイプ**として**静的コンテンツ**を選択します。



作成したリージョンにページ・アイテムを４つ作成します。

最初に画像を表示するページ・アイテム**P1_IMAGE**を作成します。リージョン**行き先**の上で**コンテキスト・メニュー**を表示させ、**ページ・アイテムの作成**を実行します。

**識別**の**名前**を**P1_IMAGE**、**タイプ**として**イメージの表示**を選択します。**ラベル**を指定すると画像に重なって表示されるため、**空白**にします。**設定**の**基準**として**Image URL stored in Page Item Value**を選択します。

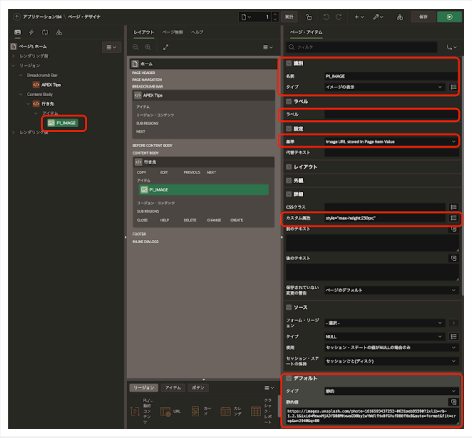
無指定だと表示される画像が大きすぎ、また、サイズもまちまちになるため、**詳細**の**カスタム属性**として以下を設定し、画像の**高さを230px**に固定します。

**style="max-height:230px;"**

**デフォルト**の**タイプ**を**静的**にします。**静的値**として以下を指定します。選択された首都の画像は、images.unsplash.comより取得し表示します。

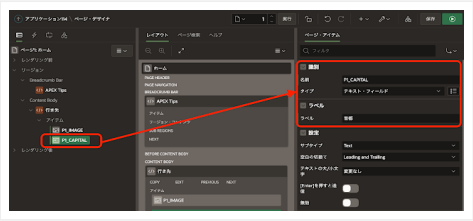
**https://images.unsplash.com/photo-1616593437252-0631aeb95590?ixlib=rb-1.2.1&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=2940&q=80**

アプリケーションの実行直後に表示される画像になります。この後、首都や国が選択されると、このURLを画面上で置き換え、表示する画像を変更します。この設定によりプレースホルダが作られるため、デフォルトの設定は必須です。



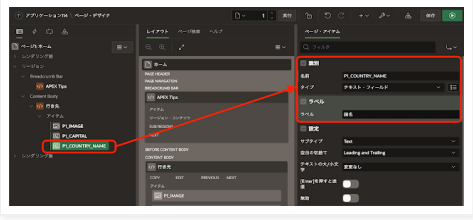
首都を保持するページ・アイテムを作成します。

**識別**の**名前**を**P1_CAPITAL**、**タイプ**は**テキスト・フィールド**とします。**ラベル**は**首都**にします。



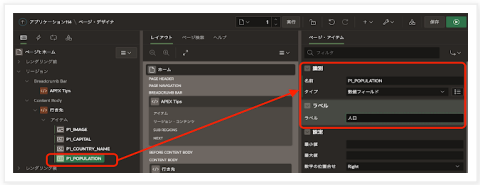
国名を保持するページ・アイテムを作成します。

**識別**の**名前**を**P1_COUNTRY_NAME**、**タイプ**は**テキスト・フィールド**とします。**ラベル**は**国名**にします。



人口を保持するページ・アイテムを作成します。

**識別**の**名前**を**P1_POPULATION**、**タイプ**は**数値フィールド**とします。**ラベル**は**人口**にします。



首都、国名、人口を、ビューEBA_COUNTRIES_Vより取り出すAJAXプロセスを作成します。

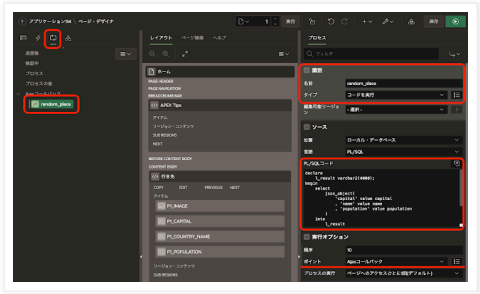
左ペインで**プロセス・ビュー**を表示させます。

**Ajaxコールバック**のノード上で**コンテキスト・メニュー**を表示させ、**プロセスの作成**を実行します。

作成したプロセスの**識別**の**名前**は**random_place**とします。**タイプ**として**コードを実行**を選択します。**ソース**の**PL/SQLコード**に以下を記載します。

```
declare
    l_result varchar2(4000);
begin
    select
        json_object(
            'success' value 'true' format json
            , 'capital' value capital
            , 'name' value name
            , 'population' value population
            returning varchar2(4000)
        )
    into
        l_result
    from eba_countries_v
    order by sys.dbms_random.random
    fetch first 1 rows only;
    sys.htp.p(l_result);
end;
```

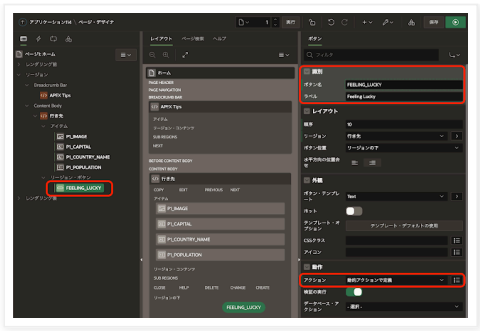
元々の実装ではパッケージAPEX_JSONを使用しています。上記ではオラクル・データベースのJSONの実装を使用しています。どちらでも結果は変わりません。



作成したAjaxコールバックを呼び出し、首都、国名、人口や画像を更新するボタンを作成します。

リージョン**行き先**上で**コンテキスト・メニュー**を表示させ、**ボタンの作成**を実行します。

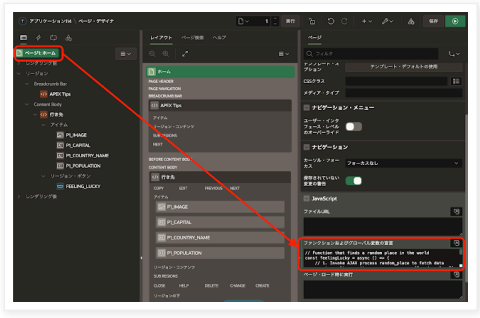
**識別**の**ボタン名**は**FEELING_LUCKY**、**ラベル**は**Feeling Lucky**とします。**動作**の**アクション**として、**動的アクションで定義**を選択します。



Ajaxコールバックrandom_placeを呼び出し、取得した情報をページ・アイテムに反映させる処理は、動的アクションに直接記述せずページのプロパティに記述します。

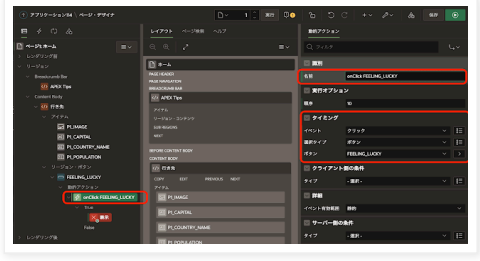
**ホーム・ページ**のプロパティの**JavaScript**、**ファンクションおよびグローバル変数の宣言**に以下を記述します。ページ・アイテムの値の設定と取得で、APEX 21.2以前でも利用可能なsetValue、getValueを使うように変更しています。

```
// Function that finds a random place in the world
const feelingLucky = async () => {
    // 1. Invoke AJAX process random_place to fetch data
    const result = await apex.server.process("random_place");
    console.log(result);

    // 2. Set display only items on the page with AJAX data
    apex.item("P1_COUNTRY_NAME").setValue(result.name);
    apex.item("P1_CAPITAL").setValue(result.capital);
    apex.item("P1_POPULATION").setValue(result.population);

    // 3. Refresh image with a random photo from unsplash
    let image_url = `https://source.unsplash.com/random/?${encodeURIComponent(apex.item("P1_CAPITAL").getValue())}`;
    console.log(image_url);
    const spinner$ = apex.util.showSpinner( apex.jQuery("#P1_IMAGE_CONTAINER") );
    apex.item("P1_IMAGE").node.onload = () => {
       spinner$.remove();
    };
    apex.item("P1_IMAGE").node.src = image_url;
};
```



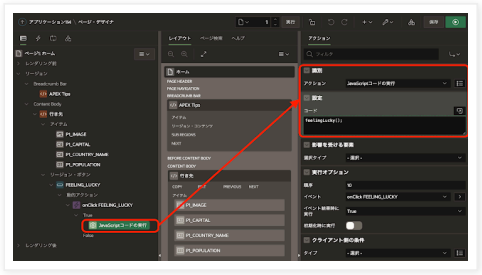
ボタンFEELING_LUCKYに、動的アクションを作成します。ボタン上で**コンテキスト・メニュー**を開き、**動的アクションの作成**を実行します。

動的アクションの**識別**の**名前**は**onClick FEELING_LUCKY**とします。ボタンに対して動的アクションを作成すると、**タイミング**は**イベント**が**クリック**、**選択タイプ**が**ボタン**、**ボタン**は**FEELING_LUCKY**となり、デフォルトがそのまま使えます。



**TRUEアクション**を選択し、**識別**の**アクション**として**JavaScriptコードの実行**を選択します。**設定**の**コード**には以下を記述します。ページ・プロパティに記載したファンクションfeelingLuckyを呼び出します。

**feelingLucky();**

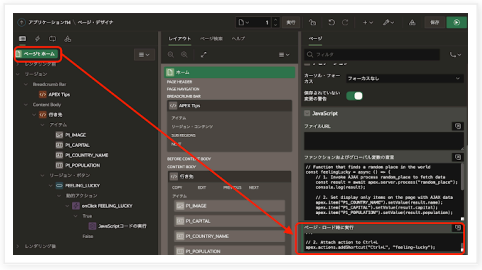


最後にキーボード・ショートカットを登録します。

ページ・プロパティの**JavaScript**の**ページ・ロード時に実行**に以下を記述します。

```
// 1. Add new action on the page that invokes a JavaScript function
apex.actions.add({
    name: "feeling-lucky",
    action: feelingLucky
});

// 2. Attach action to Ctrl+L
apex.actions.addShortcut("Ctrl+L", "feeling-lucky");
```

JavaScript APIのapex.actions.addを呼び出して、ファンクションfeelingLuckyをAPEXのアクション（apex.action）として登録します。その後にapex.actions.addShortcutを呼び出して、CTRL+Lにアクションを割り当てています。



以上でアプリケーションは完成です。実行すると先頭のGIF動画のように動作します。

今回作成したアプリケーションのエクスポートを以下に置きました。
https://github.com/ujnak/apexapps/blob/master/exports/apextipsajax.sql

この他にも色々と有用な技が紹介されているので、Office Hourの動画の視聴をお勧めします。

Oracle APEXのアプリケーション作成の参考になれば幸いです。

**完**

Yuji N. 時刻: 14:23

共有

**Yuji N.**

日本オラクル株式会社に勤務していて、Oracle APEXのGroundbreaker Advocateを拝命しました。
こちらの記事につきましては、免責事項の参照をお願いいたします。

詳細プロフィールを表示