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。

2023年8月22日火曜日

PAY.JPの支払いAPIをAPEXアプリから呼び出す

PAY.JPの支払いAPIを呼び出してカード決済を行うAPEXアプリケーションを作成してみます。

作成したAPEXアプリケーションは以下のように動作します。

PAY.JPによるカード決済を行うため、開発者アカウントを作成します。開発者アカウントを作成すると、ダッシュボードにアクセスできるようになります。また、**API設定**のページより、カード決済のテストを実施する際に必要になる、**テスト秘密鍵**と**テスト公開鍵**を取得できます。

今回の紹介する内容はPAY.JPによるカード決済を実施するために、Oracle APEX側に実装すべき最低限のコーディングになります。

Webブラウザのフロント側では、カード情報を入力するエレメントの生成および支払いAPIの呼び出しに必要なトークンの生成を実装します。

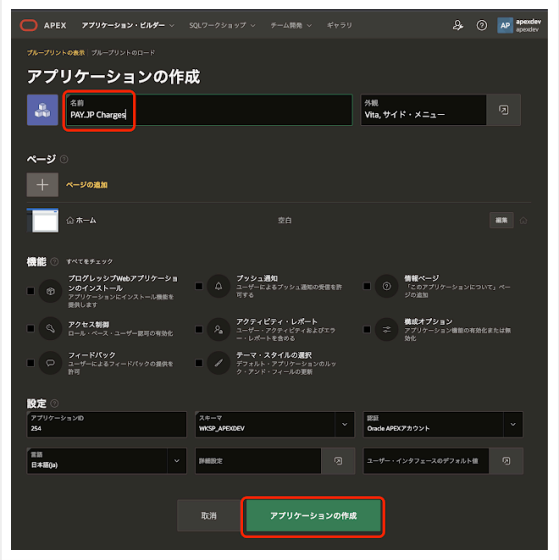
フロント側の処理はpayjp.js v2リファレンスを参照して実装します。
https://pay.jp/docs/payjs

生成したトークンと金額を受け取って、PAY.JPの支払いAPI（Charge API）を呼び出します。このREST APIは以下のリファレンスを参照して実装します。
https://pay.jp/docs/api/#charge-%E6%94%AF%E6%89%95%E3%81%84

以下より実装について説明します。

アプリケーション作成ウィザードを起動し、空のアプリケーションを作成します。**名前**は**PAY.JP Charges**とします。PAY.JPによるカード決済は、すべてホーム・ページに実装します。

アプリケーションの作成をクリックします。



アプリケーション定義の**置換文字列**として、**テスト公開鍵**と**支払いAPIのエンドポイント**を設定します。

テスト公開鍵は**置換文字列**を**PAYJP_PUBLIC_KEY**、**置換値**として**pk_test_で始まるテスト公開鍵**を設定します。

APIのエンドポイントは**置換文字列**として**PAYJP_ENDPOINT**、**置換値**として**https://api.pay.jp**を設定します。

ワークスペース・ユーティリティの**Web資格証明**を開きます。

作成済みの**Web資格証明**の一覧が表示されます。その画面で**作成**をクリックし、新たな**Web資格証明**の作成を始めます。

PAY.JPが提供するREST APIに使用するWeb資格証明は、以下の情報で作成します。

PAY.JPのAPIは、テスト秘密鍵（または本番秘密鍵）をユーザー名としたBasic認証により保護されています。Oracle APEXのWeb資格認証には**認証タイプ**として**基本認証**（Basic認証）を選択できますが、この場合、秘匿して保持されるのはパスワードに限られユーザー名は誰でも参照できます。

PAY.JPのテスト秘密鍵（もちろん本番秘密鍵も）は秘匿が必須なので、基本認証の代わりに**認証タイプ**として**HTTPヘッダー**を選択します。**資格証明名**はAuthorization、**資格証明シークレット**として、文字列**Basic**で初めて**空白**で区切り、**テスト秘密鍵の末尾に':'（コロン）をつけてBase64でエンコードした値**を設定します。

Linuxなどのコマンドラインでは、以下のような処理によって資格証明シークレットを生成できます。

echo Basic `echo sk_test_xxxxxxxxxxxxxxxxxxxxxxxx: | base64`

```
% echo Basic `echo sk_test_xxxxxxxxxxxxxxxxxxxxxxxx: | base64`
Basic cGtxxXafea*****aeaqf***********k6Cg==
%
```

Web資格証明の**名前**は**PAY.JP Test CRED**、**静的ID**は**PAYJP_TEST_CRED**として、**Web資格証明**を作成します。**URLに対して有効**は**https://api.pay.jp**を設定します。

Web資格証明はアプリケーションのエクスポートから除外されるため、秘密鍵が誤って流出することを防ぐことができます。

以上で**作成**をクリックします。

Web資格証明として**PAY.JP Test CRED**（静的IDは**PAYJP_TEST_CRED**）が作成されました。

ホーム・ページにいくつかのコンポーネントを配置します。

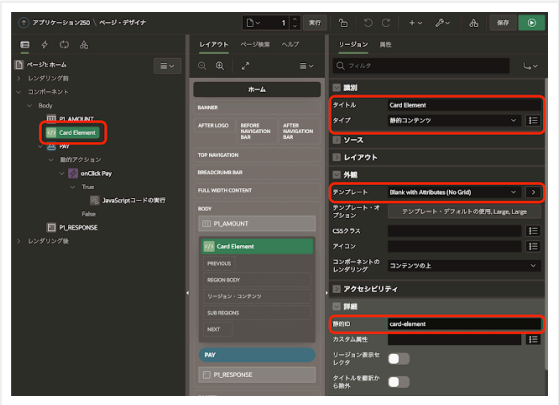
最初に決済する金額を入力するページ・アイテムを作成します。

ページ・アイテムの**識別**の**名前**は**P1_AMOUNT**、**タイプ**として**数値フィールド**を選択します。ラベ

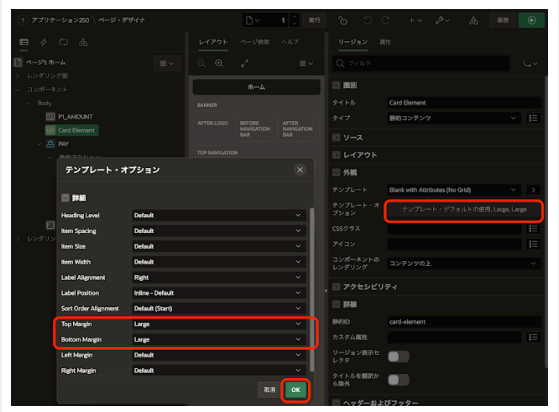
ルは**Amount**とします。

カード情報を入力するリージョンを作成します。

識別の**タイトル**は**Card Element**、**タイプ**として**静的コンテンツ**を選択します。リージョンの描画はPAY.JPより提供されているSDKによって実行するため、APEX側では余計な修飾は省きます。**外観**の**テンプレート**に**Black with Attributes (No Grid)**を選択し、**詳細**の**静的ID**として**card-element**を指定します。



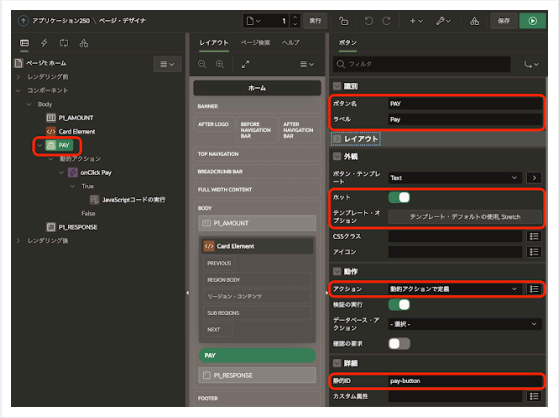
カード情報の上下にあるページ・アイテムとボタンとの間隔が狭いので、少し広げます。**テンプレート・オプション**を開き、**詳細**の**Top Margin**と**Bottom Margin**を**Large**に変更します。



支払いを実行するボタンを作成します。

識別の**ボタン名**は**PAY**、**ラベル**は**Pay**とします。**外観**の**ホット**を**オン**にします。

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。**詳細**の**静的ID**に**pay-button**を設定します。以下の画面ショットでは動的アクションが作成されていますが、これは後で作成します。



支払いAPIのレスポンスをそのまま表示するページ・アイテムを作成します。

識別の**名前**は**P1_RESPONSE**、**タイプ**として**テキスト領域**を選択します。**ラベル**は**Response**とします。

ページへ配置するコンポーネントは作成できました。これより、支払い処理を実装します。

ページ・プロパティの**JavaScript**の**ファイルURL**として以下を設定します。

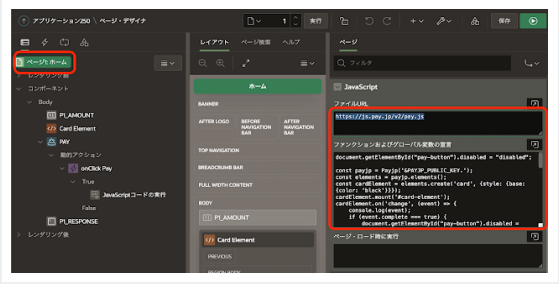
https://js.pay.jp/v2/pay.js

ファンクションおよびグローバル変数の宣言として以下を記述します。

```
document.getElementById("pay-button").disabled = "disabled";
const payjp = Payjp('&PAYJP_PUBLIC_KEY.');
const elements = payjp.elements();
const cardElement = elements.create('card', {style: {base: {color: 'black'}}});
cardElement.mount('#card-element');
cardElement.on('change', (event) => {
    console.log(event);
    if (event.complete === true) {
        document.getElementById("pay-button").disabled = null;
    } else {
        document.getElementById("pay-button").disabled = "disabled";
    }
    if (event.error !== null) {
        apex.message.clearErrors();
        apex.message.showErrors(
            {
                type:       "error",
```

```
                location:   "inline",
                pageItem:   "card-element",
                message:    event.error.message,
                unsafe:     false
            }
        );
    } else {
        let elem = document.getElementById("card-element_error_placeholder");
        if (elem !== null) {
            elem.remove();
        }
        // apex.message.clearErrors();
    };
});
```

create-element.js hosted with ❤️ by GitHub view raw



ボタンPAYに動的アクションを作成します。

動的アクションの**識別**の**名前**は**onClick Pay**、**タイミング**の**イベント**はボタンのデフォルトである**クリック**です。

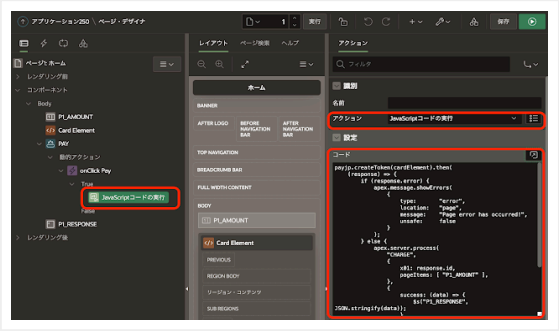
TRUEアクションとして**JavaScriptコードの実行**を選択し、**設定**の**コード**に以下を記述します。

```
paypj.createToken(cardElement).then(
    (response) => {
        if (response.error) {
            apex.message.showErrors(
                {
                    type:       "error",
                    location:   "page",
                    message:    "Page error has occurred!",
                    unsafe:     false
```

```
                }
            );
        } else {
            apex.server.process(
                "CHARGE",
                {
                    x01: response.id,
                    pageItems: [ "P1_AMOUNT" ],
                },
                {
                    success: (data) => {
                        $s("P1_RESPONSE", JSON.stringify(data));
                    },
                    error: (jqXHR, textStatus, errorThrown) => {
                        console.log(textStatus);
                    }
                }
            )
        }
    }
);
```

token-and-call.js hosted with ❤️ by GitHub view raw

最初にカード情報からトークンを生成し、そのトークンと金額を引数として、PAY.JPの支払いAPIを呼び出すAjaxコールバックを呼び出します。



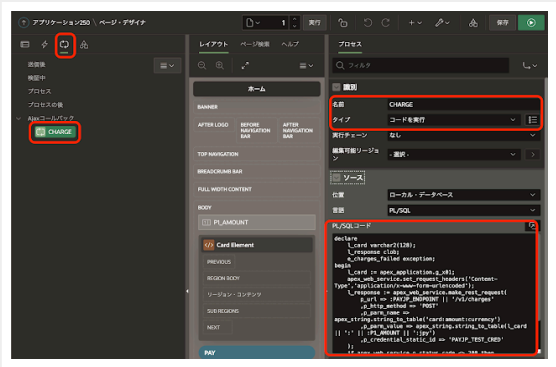
PAY.JPの支払いAPIの呼び出しはAjaxコールバックより行います。

Ajaxコールバックとして**CHARGE**を作成し、以下のコードを記述します。支払いAPIの仕様に従ってパラメータを渡し、/v1/chagesを呼び出しています。

```
declare
    l_card varchar2(128);
    l_response clob;
    e_charges_failed exception;
begin
    l_card := apex_application.g_x01;
    apex_web_service.set_request_headers('Content-Type','application/x-www-form-urlencoded');
    l_response := apex_web_service.make_rest_request(
```

```
        p_url => :PAYJP_ENDPOINT || '/v1/charges'
        ,p_http_method => 'POST'
        ,p_parm_name => apex_string.string_to_table('card:amount:currency')
        ,p_parm_value => apex_string.string_to_table(l_card || ':' || :P1_AMOUNT || ':jpy')
        ,p_credential_static_id => 'PAYJP_TEST_CRED'
    );
    if apex_web_service.g_status_code <> 200 then
        apex_debug.info(l_response);
        raise e_charges_failed;
    end if;
    htp.p(l_response);
end;
```

call-charge-api.sql hosted with ❤️ by GitHub view raw



以上でPAY.JPの支払いAPIを呼び出すサンプルは完成です。アプリケーションを実行すると、この記事の先頭のGIF動画のように動作します。

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: 18:08

共有

‹ ホーム ›

ウェブ バージョンを表示

自己紹介

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