# 日日是Oracle APEX

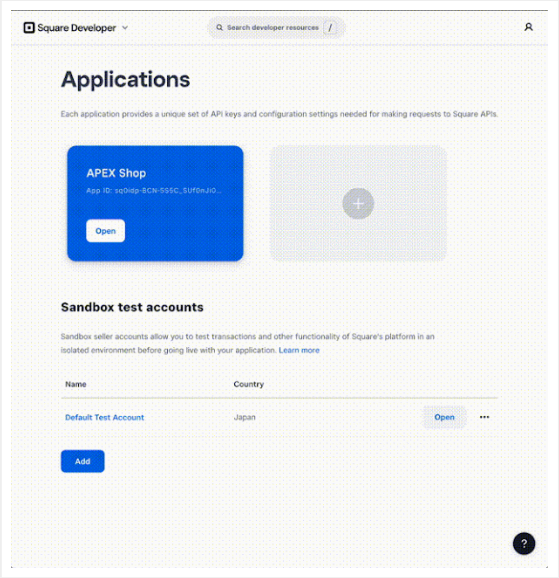
Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2023年8月21日月曜日

## SquareのPayment APIをAPEXアプリから呼び出す

SquareのPayment 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。




SquareのPayment APIを呼び出すために、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、Developer Dashboardにアクセスできるようになります。開発者アカウントを作成する際に、店舗やアプリケーションといったサンドボックス環境を使用するために必要な、最低限の設定が行われます。

Oracle APEXのアプリケーションよりSquareのPayment APIを呼び出す処理は、以下の説明を参考に実装しています。

**Take a Card Payment with the Web Payments SDK**
https://developer.squareup.com/docs/web-payments/take-card-payment

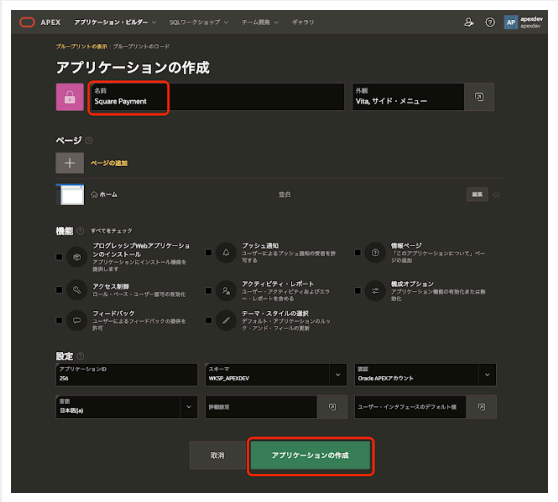
上記のではサーバー側はNode.js（JavaScript）で実装されています。PL/SQLの実装に変更するために、以下のPayment APIのリファレンスを参照しています。

**Create payment**
https://developer.squareup.com/reference/square/payments-api/create-payment

以下より実装について説明します。

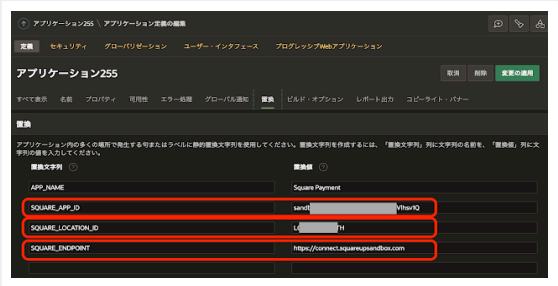
**アプリケーション作成ウィザード**を起動し、空のアプリケーションを作成します。**名前**は**Square Payment**とします。Squareで決済を行う実装は、すべてホーム・ページに行います。

**アプリケーションの作成**をクリックします。

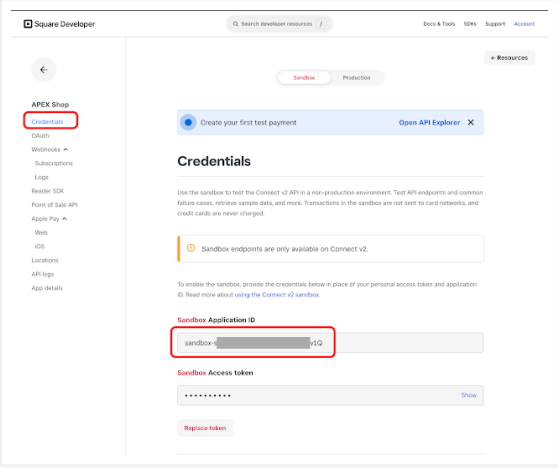



**アプリケーション定義**の**置換文字列**として、Payment APIの引数となる値を設定します。

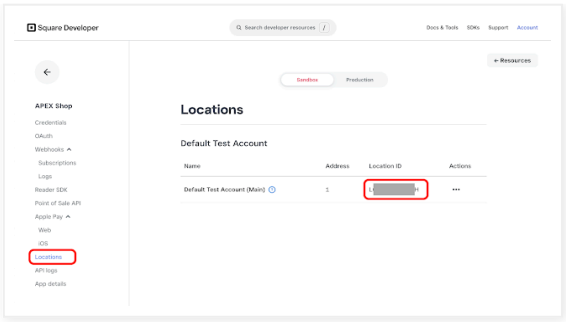
置換文字列として次の３つ、**SQUARE_APP_ID**、**SQUARE_LOCATION_ID**、**SQUARE_ENDPOINT**を設定します。




置換文字列**SQUARE_APP_ID**には、作成済みのアプリケーションの**Credentials**の**Sandbox Application ID**の値を設定します。



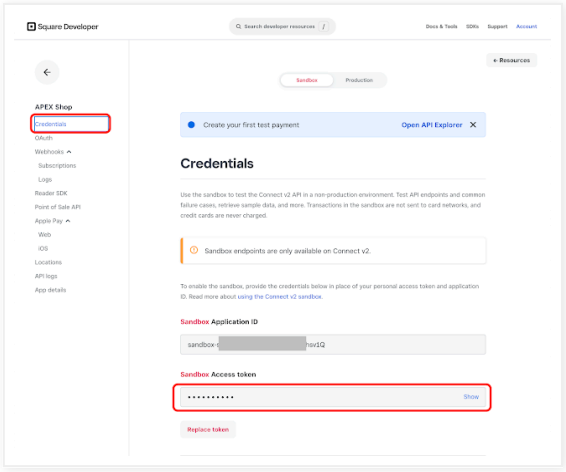

置換文字列**SQUARE_LOCATION_ID**には、**Locations**の**Location ID**を設定します。




置換文字列**SQUARE_ENDPOINT**には、サンドボックス環境を呼び出すエンドポイントを設定します。

**https://connect.squareupsandbox.com**

Payment APIを呼び出すために、**Web資格証明**を作成します。資格証明に使う値は**Sandbox Access token**です。この値を表示させ、どこかに一時保管しておきます。




**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。

Square Payment APIに使用するWeb資格証明は、以下の情報で作成します。

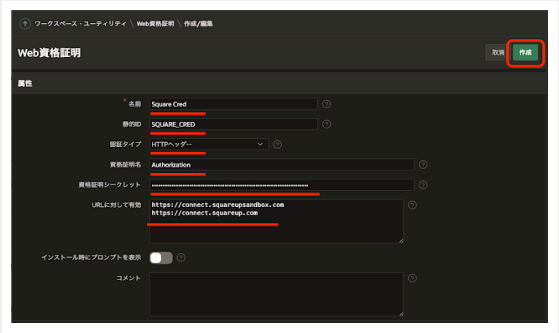
**名前**は**Square Cred**、**静的ID**は**SQUARE_CRED**とします。**認証タイプ**は**HTTPヘッダー**、**資格証明名**（つまりHTTPヘッダー名）は**Authorization**、**資格証明シークレット**として、**Bearer**で始めて**空白**で区切り、**Sandbox Access tokenを繋げた値**を設定します。

**Bearer Sandbox_Access_tokenの値**

**URLに対して有効**については、サンドボックスとプロダクションの双方のAPIのエンドポイントを設定しておきます。

**https://connect.squareupsandbox.com**
**https://connect.squareup.com**

以上で**作成**をクリックします。




**Web資格証明**として**Square Cred**（静的IDは**SQUARE_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。

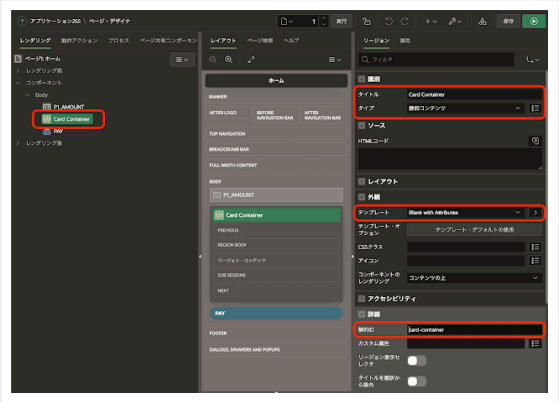
最初に決済する金額を入力するページ・アイテムを作成します。

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。

Squareのドキュメントでは決済金額を$1で決め打ちして実装しているため、このページ・アイテムに対応するコンポーネントはありません。


カード情報を入力するリージョンを作成します。

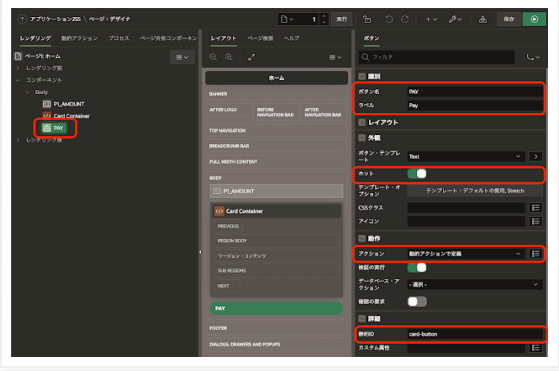
**識別**の**タイトル**は**Card Container**、**タイプ**として**静的コンテンツ**を選択します。リージョンの描画はSquareより提供されているSDKによって実行するため、APEX側では余計な修飾は省きます。**外観**の**テンプレート**に**Black with Attributes**を選択し、**詳細**の**静的ID**として**card-container**を指定します。



Payment APIを呼び出す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**を選択しますが、動的アクションは作成しません。**詳細**の**静的ID**に**card-button**を設定します。このボタンをクリックしたときに呼び出されるハンドラは、**ページ・プロパティ**の**JavaScript**の**ファンクションおよびグローバル変数の宣言**で定義します。



**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://sandbox.web.squarecdn.com/v1/square.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。

```
const appId = '&SQUARE_APP_ID.';
const locationId = '&SQUARE_LOCATION_ID.';

async function initializeCard(payments) {
    const card = await payments.card();
    await card.attach('#card-container');
    return card;
}

// Call this function to send a payment token, buyer name, and other details
// to the project server code so that a payment can be created with
// Payments API
async function createPayment(token) {
```

```
    apex.server.process(
        "CREATE_PAYMENT",
        {
            x01: token,
            x02: locationId,
            pageItems: ["P1_AMOUNT"]
        },
        {
            success: (data) => {
                return data;
            },
            error: (data) => {
                throw new Error(data);
            }
        }
    );
}

// This function tokenizes a payment method.
// The 'error' thrown from this async function denotes a failed tokenization,
// which is due to buyer error (such as an expired card). It's up to the
// developer to handle the error and provide the buyer the chance to fix
// their mistakes.
async function tokenize(paymentMethod) {
    const tokenResult = await paymentMethod.tokenize();
    if (tokenResult.status === 'OK') {
        return tokenResult.token;
    } else {
        let errorMessage = `Tokenization failed-status: ${tokenResult.status}`;
        if (tokenResult.errors) {
            errorMessage += ` and errors: ${JSON.stringify(
                tokenResult.errors
            )}`;
        }
        throw new Error(errorMessage);
    }
}

document.addEventListener('DOMContentLoaded', async function () {
    if (!window.Square) {
        throw new Error('Square.js failed to load properly');
    }
    const payments = window.Square.payments(appId, locationId);
    let card;
    try {
        card = await initializeCard(payments);
    } catch (e) {
```

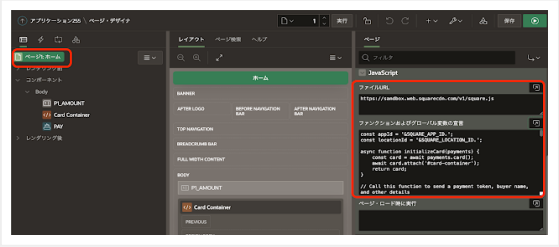
```
        console.error('Initializing Card failed', e);
        return;
    }

    // Step 5.2: create card payment
    async function handlePaymentMethodSubmission(event, paymentMethod) {
        event.preventDefault();

        try {
            // disable the submit button as we await tokenization and make a
            // payment request.
            cardButton.disabled = true;
            const token = await tokenize(paymentMethod);
            const paymentResults = await createPayment(token);
            // displayPaymentResults('SUCCESS');
            apex.message.showPageSuccess("Payment Success!")
            console.debug('Payment Success', paymentResults);
        } catch (e) {
            cardButton.disabled = false;
            // displayPaymentResults('FAILURE');
            aepx.message.clearErrors();
            apex.message.showErrors([
                {
                    type: "error",
                    location: "page",
                    message: "Payment Failed!",
                    unsafe: false
                }
            ]);
            console.error(e.message);
        }
    }

    const cardButton = document.getElementById(
        'card-button'
    );
    cardButton.addEventListener('click', async function (event) {
        await handlePaymentMethodSubmission(event, card);
    });

});
```



Squareのドキュメントにある元のコードとの違いは、大きく２点あります。

ファンクションcreatePaymentの呼び出し先をNode.jsによる実装から、Ajaxコールバックに置き替えています。そのためfetchでの呼び出しの代わりにapex.server.processを使うようにしています。

Payment APIの呼び出しの成功および失敗の表示はapex.messageを使った通知に置き替えています。そのため、IDがpayment-status-containerのDIV要素は作成せず、そのDIV要素を更新するファンクションdisplayPaymentResultsも削除しています。

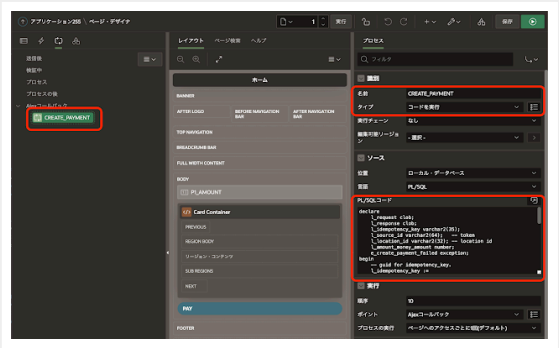
SquareのPayment APIの呼び出しはAjaxコールバックより行います。

**Ajaxコールバック**として**CREATE_PAYMENT**を作成し、以下のコードを記述します。Payment APIの仕様に従ってリクエストとなるJSONドキュメントを作成し、**/v2/payments**を呼び出しています。

```
declare
    l_request clob;
    l_response clob;
    l_idempotency_key varchar2(35);
    l_source_id varchar2(64);   -- token
    l_location_id varchar2(32); -- location id
    l_amount_money_amount number;
    e_create_payment_failed exception;
begin
    -- guid for idempotency_key.
    l_idempotency_key := regexp_replace(lower(rawtohex(sys_guid())),'(........)(....)(....)(...
    l_amount_money_amount := to_number(:P1_AMOUNT);
    l_source_id           := apex_application.g_x01;
    l_location_id         := apex_application.g_x02;
    select json_object(
        key 'source_id' value l_source_id,
        key 'location_id' value l_location_id,
        key 'idempotency_key' value l_idempotency_key,
        key 'amount_money' value json_object(
            key 'amount' value l_amount_money_amount,
            key 'currency' value 'JPY'
        )
    ) into l_request from dual;
    apex_debug.info(l_request);
    apex_web_service.clear_request_headers();
    apex_web_service.set_request_headers('Square-Version','2023-08-16',p_reset => false);
    apex_web_service.set_request_headers('Content-Type','application/json',p_reset => false);
```

```
    l_response := apex_web_service.make_rest_request(
        p_url => :SQUARE_ENDPOINT || '/v2/payments'
        ,p_http_method => 'POST'
        ,p_body => l_request
        ,p_credential_static_id => 'SQUARE_CRED'
    );
    if apex_web_service.g_status_code <> 200 then
        raise e_create_payment_failed;
    end if;
    htp.p(l_response);
end;
```

square-payment-api-backend-apex.sql hosted with ❤ by GitHub view raw



以上でSquareのPayment 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: 15:40

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