日日是Oracle APEX

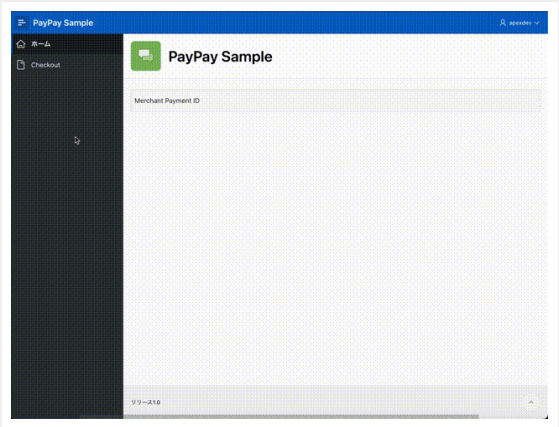
Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2023年3月1日水曜日

PayPay QRコード払いをAPEXアプリケーションに組み込む

PayPayのQRコード払いをAPEXアプリケーションに組み込んでみます。

作成するアプリケーションは以下のように動作します。スマートフォンのPayPayアプリにて画面に表示されているQRコードを読み込み、支払いを実行しています。



PayPay Open Payment APIの以下の２つのAPIを呼び出しています。

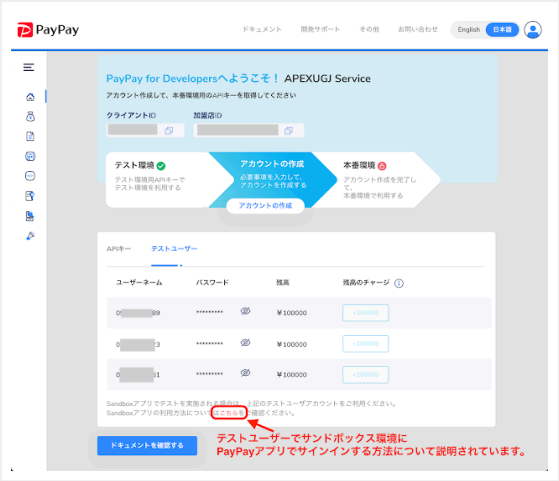
Create QR Code
https://www.paypay.ne.jp/opa/doc/jp/v1.0/webcashier#tag/%E6%B1%BA%E6%B8%88/operation/createQRCode

Get payment details
https://www.paypay.ne.jp/opa/doc/jp/v1.0/webcashier#tag/%E6%B1%BA%E6%B8%88/operation/getPaymentDetails

PayPay for Developersにアカウント登録を行い、**決済方法**として**ウェブペイメント**を登録します。

PayPay for Developersに登録すると、テスト用のサンドボックス環境が使用可能になります。この中の**加盟店ID**、**APIキー**、**APIキー・シークレット**をAPIを呼び出す際に使用します。

サンドボックス環境にはテストユーザーも用意されています。スマートフォンにインストールされたPayPayアプリより、テストユーザーでサンドボックス環境にサインインする方法については、こちらのリンク先に説明されています。



サンドボックス環境の**加盟店ID**、**APIキー**、**APIキー・シークレット**が得られていて、スマートフォンのPayPayアプリにテストユーザーでサインインできれば、PayPay側の準備は完了です。

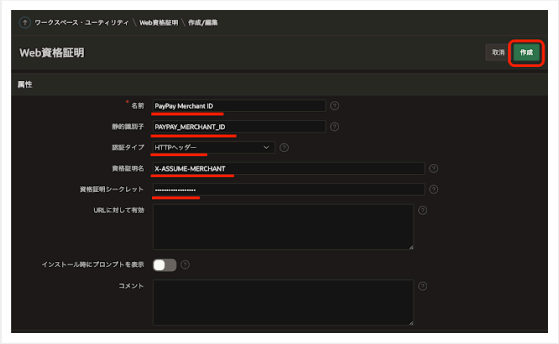
以下より、Oracle APEXのアプリケーション作成について記述します。

最初にPayPayの**加盟店ID**を**Web資格証明**として登録します。PayPayのAPI認証はHMAC認証で行われますが、これはリクエストごとに値が変わるためWeb資格証明とするのには向いていません。加盟店IDはAPI発行時に、HTTPヘッダーまたは問い合わせ文字列に必ず含める必要があるため、Web資格証明として作成することにしています。

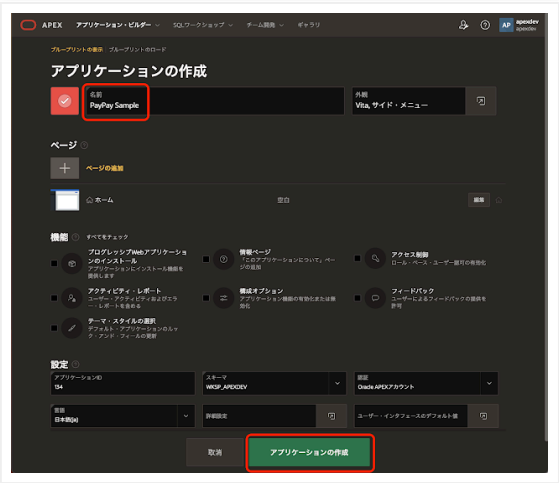
ワークスペース・ユーティリティの**Web資格証明**を開き、**作成**を実行します。

名前はPayPay Merchant IDとします。**静的識別子**はPAYPAY_MERCHANT_ID、**認証タイプ**は**HTTPヘッダー**、**資格証明名**はHTTPヘッダー名である**X-ASSUME-MERCHANT**、**資格証明シークレット**として**加盟店ID**を設定します。

作成をクリックして、Web資格証明の作成を完了します。



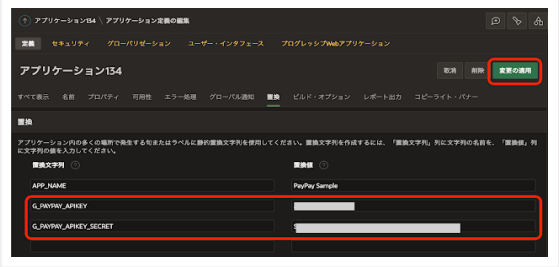
アプリケーション作成ウィザードを起動し、空のアプリケーションを作成します。**名前**は**PayPay Sample**とします。



アプリケーションが作成されたら**アプリケーション定義**を開き、置換文字列としてAPIキーとAPIキー・シークレットを設定します。これは安全な方法とはいえないため、サンドボックス環境ではなく本番環境での利用の際は、より安全な方法で保存すべきです。例えばOracle Cloudであれば、ボールト・サービスのシークレットして保存するといった実装が可能です。Oracle APEXのアプリケーションは、セッション開始時にボールトからAPIキーやシークレットを取得するようにできます。

今回はサンドボックス環境での使用を前提として、**置換文字列**を定義します。置換文字列**G_PAYPAY_APIKEY**に**APIキー**の値、**G_PAYPAY_APIKEY_SECRET**に**APIキー・シークレットの**値を設定します。

変更の適用をクリックします。

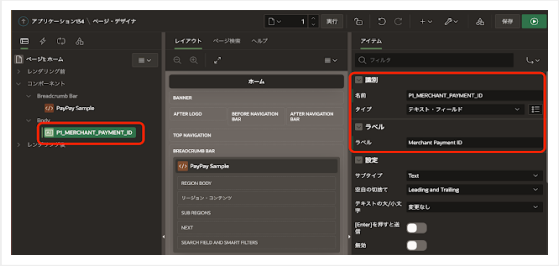


ページ・デザイナで**ホーム・ページ**を開きます。

ホーム・ページに、取引の結果を表示します。

加盟店側で設定する取引IDを保持するページ・アイテムを作成します。

ページ・アイテムの**識別**の**名前**を**P1_MERCHANT_PAYMENT_ID**、**タイプ**を**テキスト・フィールド**とします。**ラベル**は**Merchant Payment ID**とします。



P1_MERCHANT_PAYMENT_IDに対応した取引の状況をPayPayに問い合わせ、その結果を表示するリージョンを作成します。

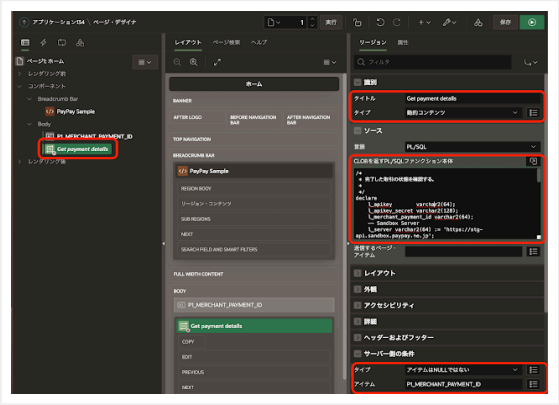
リージョンの**識別**の**タイトル**は**Get payment details**とします。**タイプ**として**動的コンテンツ**を選択します。**ソース**の**CLOBを返すPL/SQLファンクション本体**として以下を記述します。PayPayのGet payment details APIを呼び出し、応答であるJSON文書をそのまま表示します。

```
/*
 * 完了した取引の状態を確認する。
 *
 */
declare
    l_apikey        varchar2(64);
    l_apikey_secret varchar2(128);
    l_merchant_payment_id varchar2(64);
    -- Sandbox Server
    l_server varchar2(64) := 'https://stg-api.sandbox.paypay.ne.jp';
    l_request_url varchar2(80);
    l_header varchar2(200);
    l_response_clob clob;
begin
    /* APIキーとAPIキー・シークレットを取得する。 */
    l_apikey := v('G_PAYPAY_APIKEY');
    l_apikey_secret := v('G_PAYPAY_APIKEY_SECRET');

    /*
     * Get payment detailsのREST APIを発行する。
     *
     * 参照: https://www.paypay.ne.jp/opa/doc/jp/v1.0/webcashier#tag/%E6%B1%BA%E6%B8%88/operatio
     */
    l_request_url := '/v2/codes/payments/' || :P1_MERCHANT_PAYMENT_ID;
    l_header := util_paypay_api.generate_hmac_auth_header(
        p_request_url => l_request_url
        ,p_http_method => 'GET'
        ,p_apikey      => l_apikey
        ,p_apikey_secret => l_apikey_secret
```

```
    );
    apex_web_service.clear_request_headers;
    apex_web_service.set_request_headers('Authorization', l_header, p_reset => false);
    l_response_clob := apex_web_service.make_rest_request(
        p_url => l_server || l_request_url
        ,p_http_method => 'GET'
        ,p_credential_static_id => 'PAYPAY_MERCHANT_ID'
    );
    if apex_web_service.g_status_code <> 200 then
        raise_application_error(-20001, 'GET_PAYMENT_DETAILS Error = ' || apex_web_service.g_st
    end if;

    /*
     * 応答のJSONをそのまま出力する。
     */
    return l_response_clob;
end;
```

get-payment-details.sql hosted with ❤️ by GitHub view raw

サーバー側の条件の**タイプ**に**アイテムはNULLではない**を選択し、**アイテム**として**P1_MERCHANT_PAYMENT_ID**を指定します。P1_MERCHANT_PAYMENT_IDの値はPayPayの支払いが完了したときに、PayPayから制御がAPEXアプリに戻される際に設定されます。つまり、PayPayでの支払いが完了したときに、この動的リージョンのPL/SQLコードが実行されます。



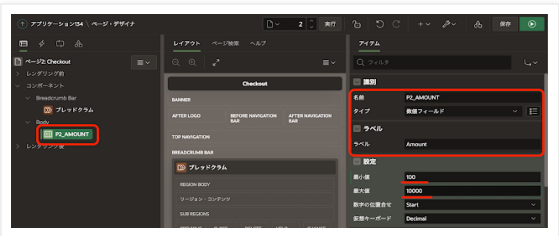
PayPayによる支払いを行うページを作成します。

ページの作成を実行し、**空白のページ**をページ番号**2**として作成します。**名前**はCheckout、**ページ・モード**は**標準**です。**ナビゲーション**はデフォルトのまま、**ブレッドクラムの使用**、**ナビゲーションの使用**ともに**ON**とします。

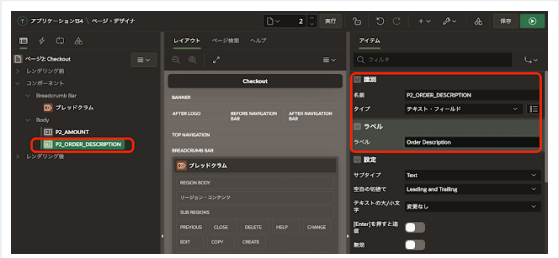
ページの作成を実行します。

作成された空白のページにページ・アイテムとボタンを作成します。

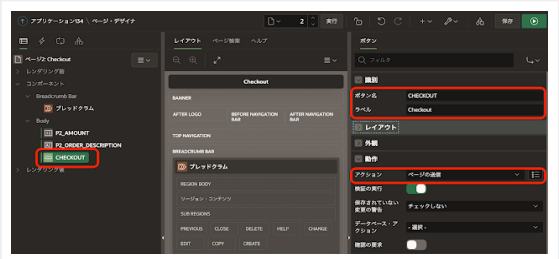
支払い金額を入力するページ・アイテム**P2_AMOUNT**を作成します。**タイプ**は**数値フィールド**、**ラベル**は**Amount**とします。必須ではありませんが、**設定**の**最小値**に**100**、**最大値**に**10000**を設定しています。



支払いの詳細を記述するページ・アイテム**P2_ORDER_DESCRIPTION**を作成します。**タイプ**は**テキスト・フィールド**、**ラベル**は**Order Description**とします。



チェックアウトを実行するボタン**CHECKOUT**を作成します。**動作**の**アクション**は**ページの送信**です。



左ペインでプロセス・ビューを開き、PayPayによる支払いを呼び出すプロセスを作成します。

作成したプロセスの**名前**は**Checkout**とします。**タイプ**として**コードの実行**を選択し、**ソース**の**PL/SQLコード**として以下を記述します。

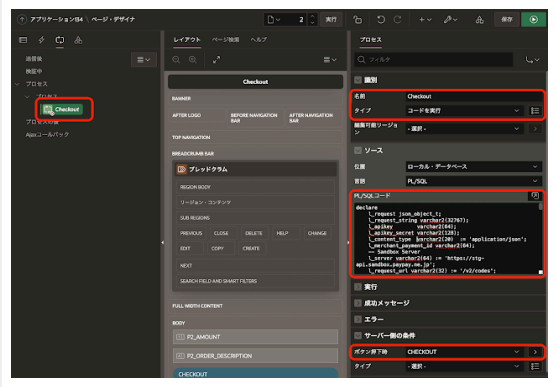
```
declare
    l_request json_object_t;
    l_request_string varchar2(32767);
    l_apikey        varchar2(64);
    l_apikey_secret varchar2(128);
    l_content_type  varchar2(20)  := 'application/json';
    l_merchant_payment_id varchar2(64);
    -- Sandbox Server
    l_server varchar2(64) := 'https://stg-api.sandbox.paypay.ne.jp';
    l_request_url varchar2(32) := '/v2/codes';
    l_header varchar2(200);
    l_amount json_object_t;
    l_redirect_url varchar2(400);
    l_user_agent   varchar2(400);
    --
    l_response_clob clob;
    l_response json_object_t;
    l_data     json_object_t;
    l_paypay_url varchar2(400);
begin
    /* APIキーとAPIキー・シークレットを取得する。 */
    l_apikey := v('G_PAYPAY_APIKEY');
    l_apikey_secret := v('G_PAYPAY_APIKEY_SECRET');
    /*
     * PayPayに送信するリクエストを作成する。
     */
    l_request := json_object_t();
    /*
     * merchantPaymentId - 加盟店から提供された一意の支払い取引ID
     *
     * データベースに取引を保存する表を作成し、その主キーの値を
     * merchantPaymentIdとする実装が一般的でしょう。
     */
    l_merchant_payment_id := sys_guid();
    l_request.put('merchantPaymentId', l_merchant_payment_id);
    /* amount - 支払金額 */
    l_amount := json_object_t();
    l_amount.put('amount', to_number(:P2_AMOUNT));
    l_amount.put('currency','JPY');
    l_request.put('amount',l_amount);
    /* orderDescription - 注文内容の説明。 */
    l_request.put('orderDescription', :P2_ORDER_DESCRIPTION);
    /* codeType - 常にORDER_QR */
    l_request.put('codeType','ORDER_QR');
    /* redirectUrl - 支払い完了後に開くページ/アプリのURL */
    l_redirect_url := apex_util.host_url || apex_page.get_url(
        p_page => 1
```

```
    ,p_items => 'P1_MERCHANT_PAYMENT_ID'
    ,p_values => l_merchant_payment_id
);
l_request.put('redirectUrl', l_redirect_url);
/* redirectType - つねにWEB_LINK */
l_request.put('redirectType','WEB_LINK');
/* userAgent  - トランザクションの発生元であるWebブラウザーのUser Agent */
l_user_agent := owa_util.get_cgi_env('HTTP_USER_AGENT');
l_request.put('userAgent', l_user_agent);
l_request_string := l_request.to_string();
-- apex_debug.info(l_request_string);

/*
 * Create QR CodeのREST APIを発行する。
 *
 * 参照: https://www.paypay.ne.jp/opa/doc/jp/v1.0/webcashier#tag/%E6%B1%BA%E6%B8%88/operatio
 */
l_header := util_paypay_api.generate_hmac_auth_header(
    p_request_body => l_request_string
    ,p_content_type => l_content_type
    ,p_request_url => l_request_url
    ,p_http_method => 'POST'
    ,p_apikey      => l_apikey
    ,p_apikey_secret => l_apikey_secret
);
apex_web_service.clear_request_headers;
apex_web_service.set_request_headers('Content-Type', l_content_type, p_reset => false);
apex_web_service.set_request_headers('Authorization', l_header, p_reset => false);
l_response_clob := apex_web_service.make_rest_request(
    p_url => l_server || l_request_url
    ,p_http_method => 'POST'
    ,p_body => l_request_string
    ,p_credential_static_id => 'PAYPAY_MERCHANT_ID'
);
-- apex_debug.info(l_response_clob);
if apex_web_service.g_status_code <> 201 then
    raise_application_error(-20001, 'ORDER_QR Error = ' || apex_web_service.g_status_code);
end if;

/*
 * レスポンスからurlを取り出し、PayPayの決済画面に遷移する。
 */
l_response := json_object_t(l_response_clob);
l_data := treat(l_response.get('data') as json_object_t);
l_paypay_url := l_data.get_string('url');
apex_util.redirect_url(
    p_url => l_paypay_url
```

```
    );
end;
```

paypay-checkout.sql hosted with ❤️ by GitHub view raw

サーバー側の条件の**ボタン押下時**に**CHECKOUT**を指定し、ボタン**CHECKOUT**が押された時に実行されるようにします。



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

今回作成したAPEX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/paypay-sample.zip

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: 15:55

共有

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