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24日木曜日

StripeのPaymentIntentsを使ったカード決済をAPEXアプリケーションに実装する

Stripeによる決済の実装は、以前にノーコードについては紹介したことがあります。こちらの記事になります。今回はPaymentIntentsを使ってカード決済を実装してみます。

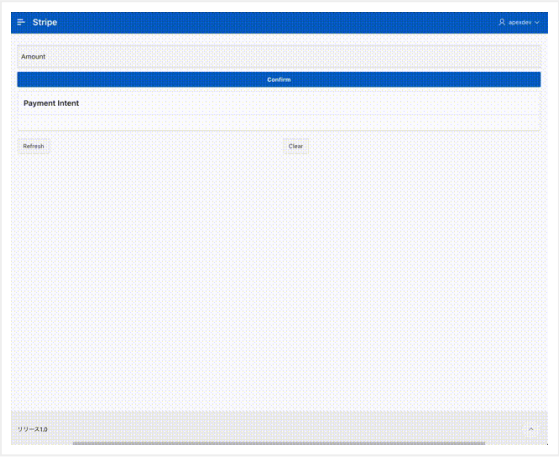
Stripeの以下のドキュメントの記述にそって、Oracle APEXに合わせた実装を行います。

支払いページを作成する
https://stripe.com/docs/connect/creating-a-payments-page?platform=web&ui=elements&locale=ja-JP

Payment Intentの作成には、以下のREST APIを呼び出します。
https://stripe.com/docs/api/payment_intents/create

カード番号の入力フォームはPayment Elementとして説明されています。
https://stripe.com/docs/js/element/payment_element

作成したアプリケーションは以下のように動作します。



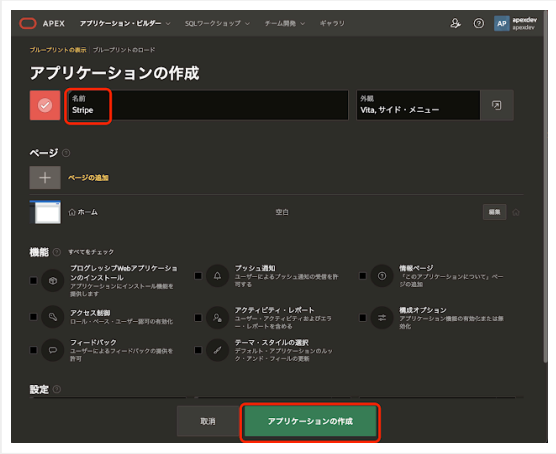
最後にREST APIを呼び出してPayment Intentを表示しています。実際に決済の成功や失敗を確認するにはWebhookを実装するのが一般的だと思いますが、本記事ではWebhookの作成は取り扱っていません。通常Webhookは、APEXのアプリケーションではなく、ORDSのRESTサービスとして実装します。

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Stripeと

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、**公開可能キー**と**APIのエンドポイント**を設定します。

パブリックキーは**置換文字列**を**STRIPE_PUBLISHABLE_KEY**、**置換値**として**pk_test_で始まる（テスト用の）公開可能キー**を設定します。

APIのエンドポイントは**置換文字列**として**STRIPE_ENDPOINT**、**置換値**として**https://api.stripe.com**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。

Stripeが提供するREST APIに使用するWeb資格証明は、以下の情報より作成します。

StripeのAPIは、ベアラーとして**シークレットキー**を渡すことで認証できます。**認証タイプ**に**HTTPヘッダー**を選択し、**資格証明名**は**Authorization**、**資格証明シークレット**として、文字列**Bearer**で始めて**空白**で区切り、**シークレットキー**の値を設定します。

Bearer シークレットキー

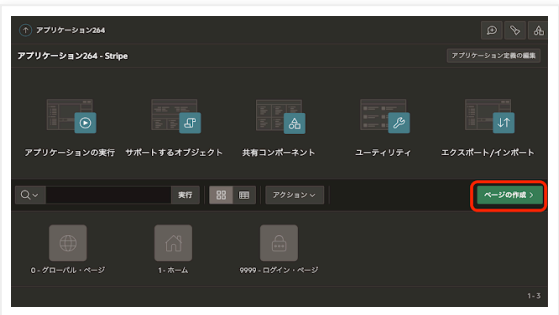
Web資格証明の**名前**は**Stripe CRED**、**静的ID**は**STRIPE_CRED**として、**Web資格証明**を作成します。
URLに対して有効は**https://api.stripe.com**を設定します。

以上で**作成**をクリックします。

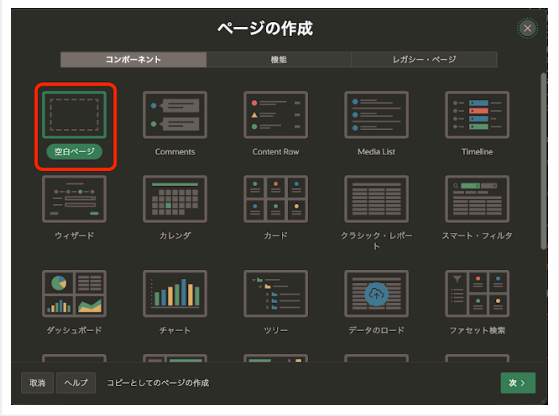
Web資格証明としてStripe CRED（静的IDは**STRIPE_CRED**）が作成されました。

Stripeによるカード情報入力フォームを表示するページを作成します。

ページの作成をクリックします。



空白ページを選択します。



ページ番号を**2**（ページ・アイテム名に現れるため２とします）、**名前**を**Pay**、**ページ・モード**として**ドロワー**を選択します。

ページの作成をクリックします。

ページPayが作成されました。

ホーム・ページでは決済金額を入力し、確定ボタンを押します。確定ボタンを押すとStripeが提供するカード情報を入力するフォームを実装したドロワーが開き、そこで決済を実行します。決済を実行すると、ドロワーが閉じます。

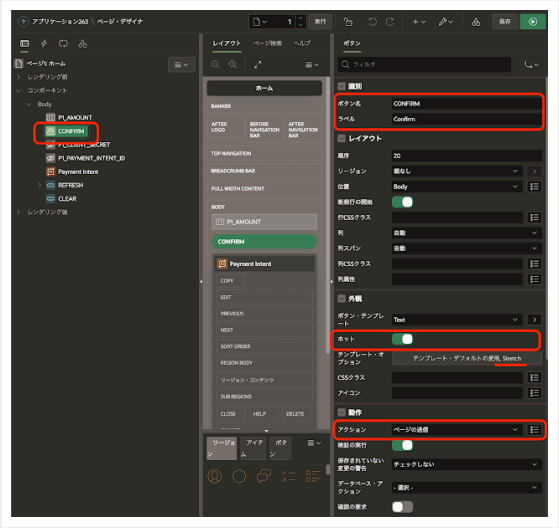
ページ・デザイナで**ホーム・ページ**を開きます。

金額を入力するページ・アイテム**P1_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。

決済金額を確定するボタン**CONFIRM**を作成します。

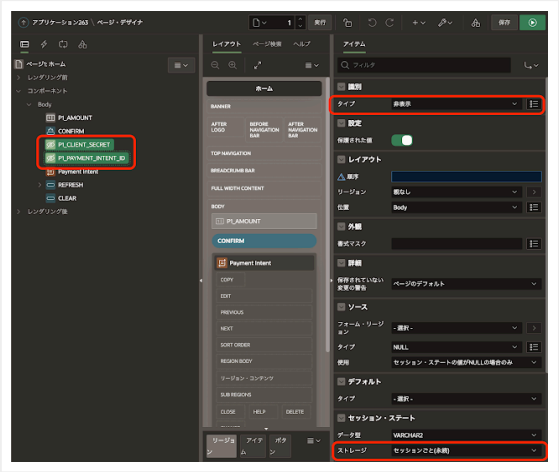
ラベルは**Confirm**、**外観**の**ホット**を**オン**、**テンプレート・オプション**の**Width**として**Stretch**を選択します。**動作**の**アクション**はデフォルトの**ページの送信**から変更しません。



作成した**Payment Intent ID**を保持するページ・アイテム**P1_PAYMENT_INTENT_ID**と**Client Secret**を保持するページ・アイテム**P1_CLIENT_SECRET**を作成します。双方とも**タイプ**は**非表示**です。

Payment Intent IDはPayment Intentを表示する際に使用し、決済の確定では使用しません。APIリファレンスには、シークレットキーとPayment Intent IDを組み合わせてPayment Intentを取得する、または、公開可能キーとClient Secretを組み合わせてPayment Intentを取得する２通りの手順があるとのことです。Client Secretは決済の確定に使用します。

セッション・ステートの**ストレージ**として**セッションごと(永続)**を選択します。決済の確定はPL/SQLで実行します。その際にページ・アイテムに設定された**Payment Intent ID**は、ページ遷移の後でも値を維持するようにします。



Payment Intentを表示するリージョンを作成します。

識別の**タイトル**は**Payment Intent**、**タイプ**として**動的コンテンツ**を選択します。

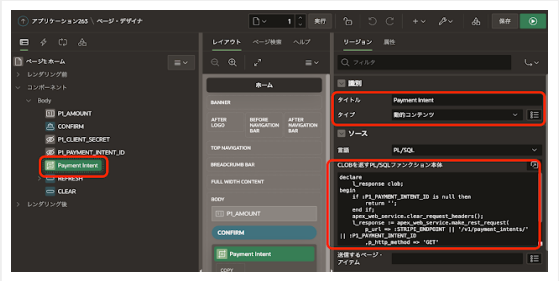
ソースのCLOBを返すPL/SQLファンクション本体として以下を記述します。

```
declare
    l_response clob;
begin
    if :P1_PAYMENT_INTENT_ID is null then
        return '';
```

```
    end if;
    apex_web_service.clear_request_headers();
    l_response := apex_web_service.make_rest_request(
        p_url => :STRIPE_ENDPOINT || '/v1/payment_intents/' || :P1_PAYMENT_INTENT_ID
        ,p_http_method => 'GET'
        ,p_credential_static_id => 'STRIPE_CRED'
    );
    return l_response;
end;
```

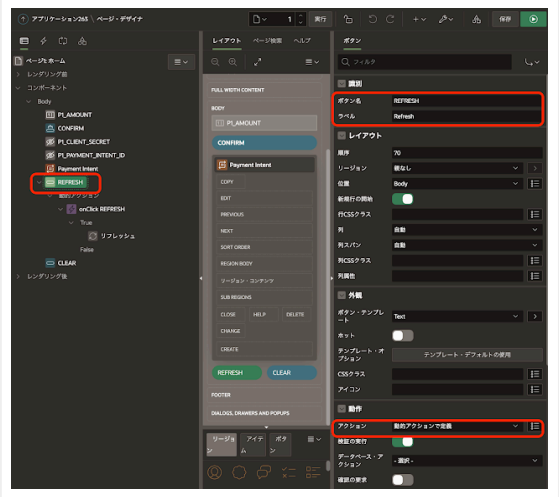
get-payment-intent.sql hosted with ❤️ by GitHub view raw

ページ・アイテム**P1_PAYMENT_INTENT_ID**はセッション・ステートとして保持されているため、**送信するページ・アイテム**への設定は**不要**です。



このリージョンをリフレッシュするボタン**REFRESH**を作成します。

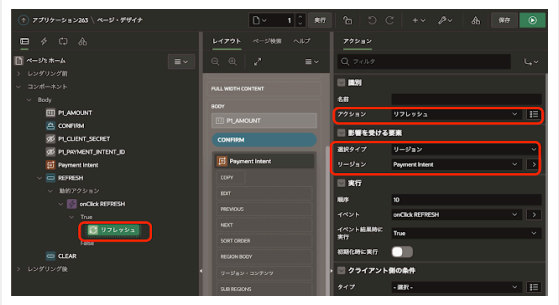
ラベルは**Refresh**、**動作**の**アクション**として**動的アクションで定義**を選択します。



ボタン**REFRESH**の**動的アクション**を作成します。

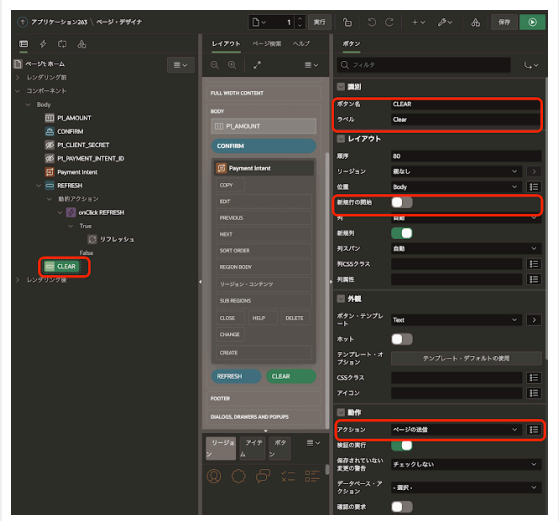
識別の**名前**は**onClick REFRESH**、**タイミング**の**イベント**はボタンのデフォルトである**クリック**です。

TRUEアクションとして**リフレッシュ**を選択します。**影響を受ける要素**の**選択タイプ**は**リージョン**、**リージョン**として**Payment Intent**を選択します。



このページを初期化するボタン**CLEAR**を作成します。

ラベルは**Clear**、**動作**の**アクション**は**ページの送信**です。**レイアウト**の**新規行の開始**は**オフ**にし、ボタンREFRESHの右隣に配置します。



プロセス・ビューを開き、ボタンを押した時に実行されるプロセスを作成します。

ボタンCONFIRMを押した時にPayment Intentを作成します。

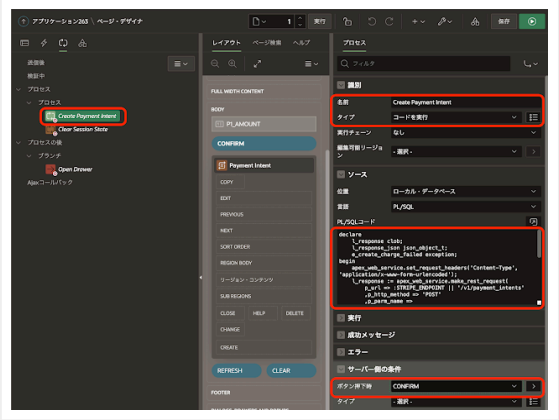
プロセスの**識別**の**名前**は**Create Payment Intent**、**タイプ**として**コードを実行**を選択します。**ソース**の**PL/SQLコード**として以下を記述します。Payment Intentを作成するための最低限の処理です。

```
declare
    l_response clob;
    l_response_json json_object_t;
    e_create_charge_failed exception;
```

```
begin
    apex_web_service.set_request_headers('Content-Type', 'application/x-www-form-urlencoded');
    l_response := apex_web_service.make_rest_request(
        p_url => :STRIPE_ENDPOINT || '/v1/payment_intents'
        ,p_http_method => 'POST'
        ,p_parm_name => apex_util.string_to_table('amount:currency:automatic_payment_methods[en
        ,p_parm_value => apex_util.string_to_table(:P1_AMOUNT || ':jpy:true')
        ,p_credential_static_id => 'STRIPE_CRED'
    );
    apex_debug.info(l_response);
    if apex_web_service.g_status_code <> 200 then
        apex_debug.info(l_response);
        raise e_create_charge_failed;
    end if;
    l_response_json := json_object_t(l_response);
    :P1_PAYMENT_INTENT_ID := l_response_json.get_string('id');
    :P1_CLIENT_SECRET := l_response_json.get_string('client_secret');
end;
```

stripe-create-payment-intent.sql hosted with ❤️ by GitHub　view raw

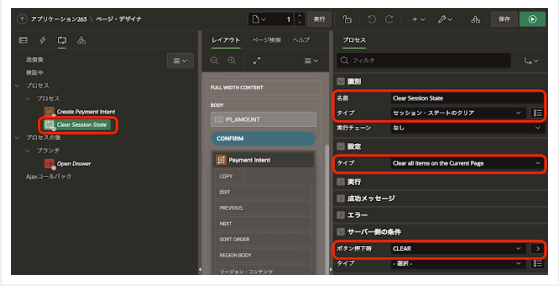
サーバー側の条件の**ボタン押下時**として**CONFIRM**を設定します。



ページを初期化するプロセスを作成します。

識別の**名前**は**Clear Session State**、**タイプ**は**セッション・ステートのクリア**、**設定**の**タイプ**として**Clear all Items on the Current Page**を選択します。

サーバー側の条件の**ボタン押下時**として**CLEAR**を設定します。

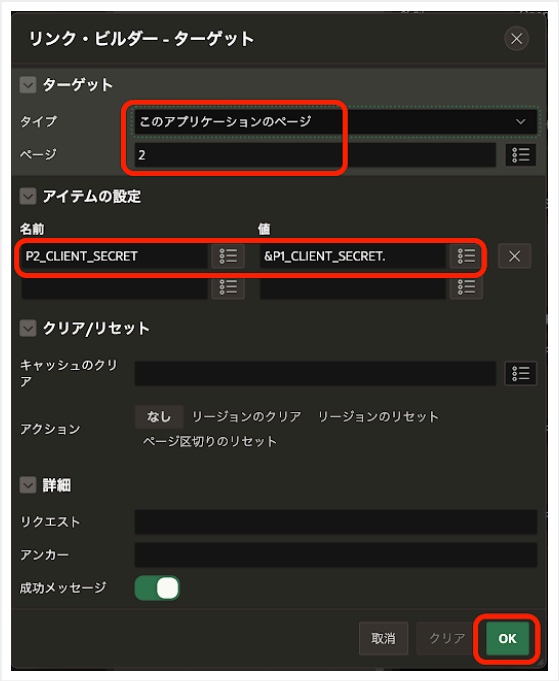


ブランチを作成し**プロセスの後**に実行するように配置します。

識別の**名前**は**Open Drawer**、**動作**の**タイプ**として**ページまたはURL(リダイレクト)**を選択します。

サーバー側の条件の**ボタン押下時**として**CONFIRM**を設定します。

ターゲットは、**タイプ**が**このアプリケーションのページ**、**ページ**は**2**です。**アイテムの設定**として、ページ・アイテム**P2_CLIENT_SECRET**に**&P1_CLIENT_SECRET.**の値が渡るようにします。



以上で、ボタンをCONFIRMをクリックするとPayment Intentが作成され、そのClient Secretを（ページ・アイテムP2_CLIENT_SECRETへの）引数としてドロワーが開くようになります。

ページ・デザイナでページ**Pay**を開きます。

ページ・プロパティの**JavaScript**の**ファイルURL**として、Stripeが提供しているJavaScriptライブラリを指定します。

https://js.stripe.com/v3/

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。

```
const stripe = Stripe('&STRIPE_PUBLISHABLE_KEY.');
const elements = stripe.elements();
```

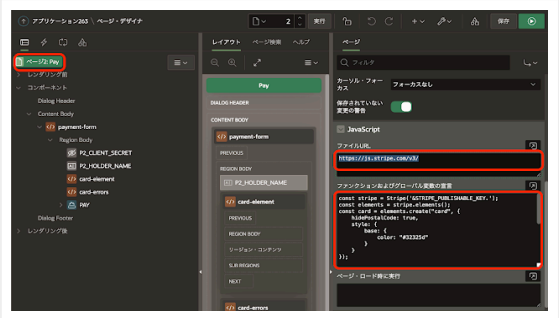
```
const card = elements.create("card", {
    hidePostalCode: true,
    style: {
        base: {
            color: "#32325d"
        }
    }
});

document.addEventListener('DOMContentLoaded', async function () {
    card.mount("#card-element");

    card.on('change', ({error}) => {
        let displayError = document.getElementById('card-errors');
        if (error) {
            displayError.textContent = error.message;
        } else {
            displayError.textContent = '';
        }
    });
});
```

stripe-card-element.js hosted with ❤️ by GitHub view raw

カード情報の入力フォームを生成しています。



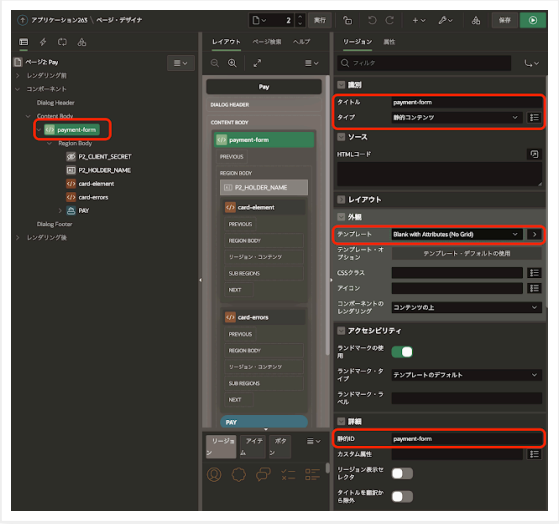
クレジットカード情報を入力するフォームを生成するリージョンを作成します。

Content Bodyの直下に**タイプ**が**静的コンテンツ**のリージョンを作成します。

識別の**タイトル**は**payment-form**、余計な装飾を省くため**外観**の**テンプレート**として**Blank with Attributes (No Grid)**を選択します。**詳細**の**静的ID**として**payment-form**を設定します。このリージョンは実施的に以下のDIV要素を生成します。

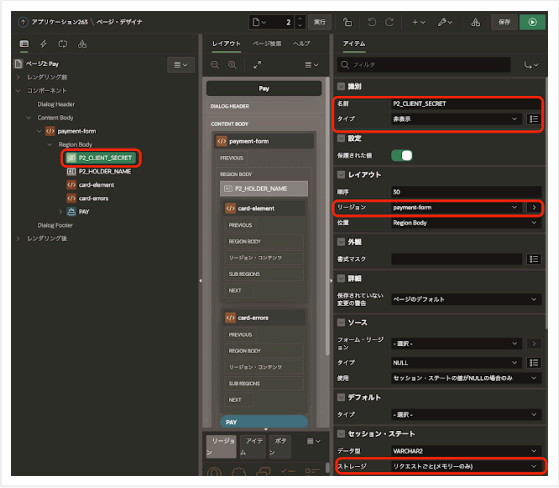
** .... **

Stripeのチュートリアルではidがpayment-formの要素はFORMです。Oracle APEXが生成したページにFORM要素がすでに含まれているためか、<form id="payment-form">では動作しません。そのため、代わりにDIV要素を作成しています。



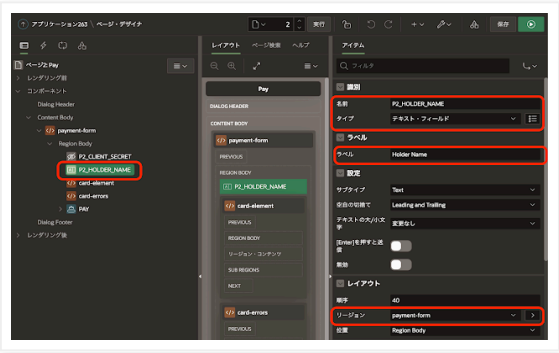
リージョン**payment-form**に**Client Secret**を保持するページ・アイテムを作成します。

識別の**名前**は**P2_CLIENT_SECRET**、**タイプ**として**非表示**を選択します。**レイアウト**の**リージョン**として**payment-form**を選択します。**セッション・ステート**の**ストレージ**は**リクエストごと(メモリーのみ)**とします。



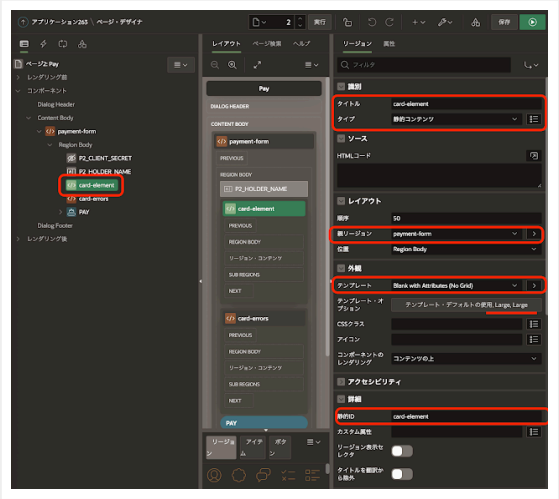
クレジット・カードの所有者を入力するページ・アイテムを作成します。

識別の**名前**は**P2_HOLDER_NAME**、**タイプ**は**テキスト・フィールド**、**ラベル**は**Holder Name**とします。**レイアウト**の**リージョン**として**payment-form**を選択します。

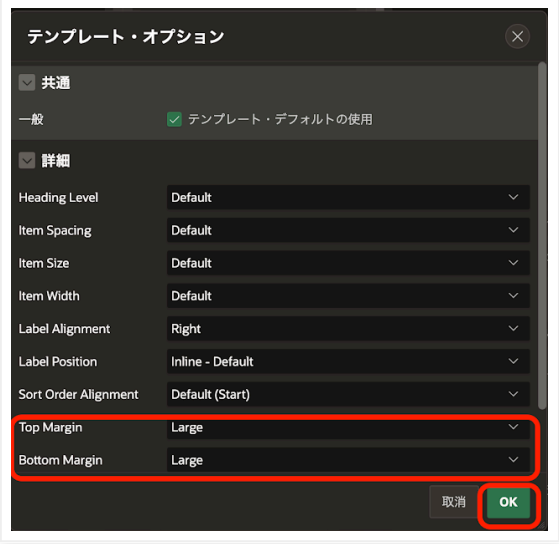


リージョン**payment-form**の子リージョンを作成します。

識別の**タイトル**はcard-element、**タイプ**は**静的コンテンツ**です。**レイアウト**の**親リージョン**として**payment-form**を指定します。このリージョンも余計な装飾を省くため、**外観**の**テンプレート**として**Blank with Attributes (No Grid)**を選びます。**詳細**の**静的ID**として**card-element**を設定します。

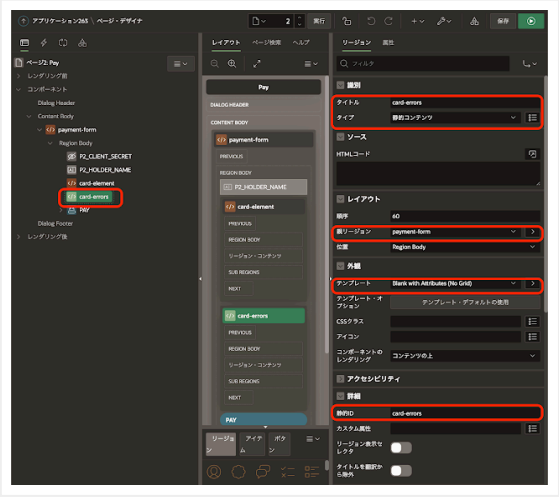


上にあるカードの所有者を入力するページ・アイテムと下にあるボタンが、カード情報を入力するリージョンに近過ぎたため、**テンプレート・オプション**の**Top Margin**と**Bottom Margin**を**Large**に変更しています。



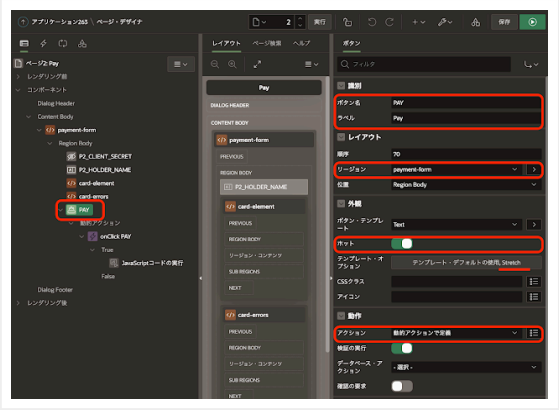
エラーを表示するリージョンを作成します。

識別の**タイトル**はcard-errors、**タイプ**は**静的コンテンツ**です。**レイアウト**の**親リージョン**として**payment-form**を指定します。このリージョンも余計な装飾を省くため、**外観**の**テンプレート**として**Blank with Attributes (No Grid)**を選びます。**詳細**の**静的ID**として**card-errors**を設定します。



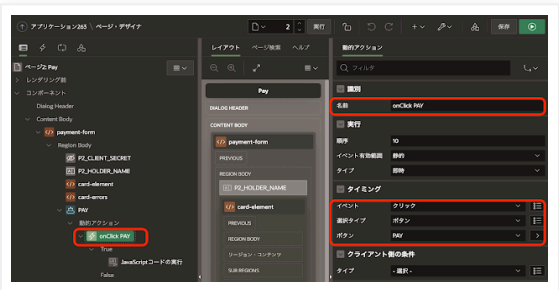
決済を確定するボタン**PAY**を作成します。

ラベルは**Pay**、**レイアウト**の**リージョン**として**payment-form**、**外観**の**ホット**を**オン**にし、**テンプレート・オプション**の**Width**に**Stretch**を選択します。**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。



ボタン**PAY**に動的アクションを作成します。

識別の**名前**は**onClick PAY**とします。**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。



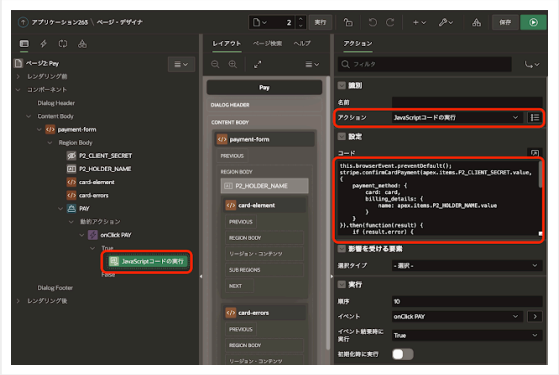
TRUEアクションとして**JavaScriptコードの実行**を選択し、**設定**の**コード**に以下を記述します。

```
this.browserEvent.preventDefault();
stripe.confirmCardPayment(apex.items.P2_CLIENT_SECRET.value, {
    payment_method: {
        card: card,
```

```
        billing_details: {
            name: apex.items.P2_HOLDER_NAME.value
        }
    }
}).then(function(result) {
    if (result.error) {
        // Show error to your customer (for example, insufficient funds)
        console.log(result.error.message);
    } else {
        // The payment has been processed!
        if (result.paymentIntent.status === 'succeeded') {
            apex.navigation.dialog.close(true);
            // Show a success message to your customer
            // There's a risk of the customer closing the window before callback
            // execution. Set up a webhook or plugin to listen for the
            // payment_intent.succeeded event that handles any business critical
            // post-payment actions.
        }
    }
});
```

stripe-confirm-card-payment.js hosted with ❤️ by GitHub　view raw

カード決済が確定するとドロワーを閉じます。



以上でStripeのPayment Intentを使ったの決済サービスの実装は完了です。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: 17:45

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