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。

2021年2月26日金曜日

ページ・アイテムの受け渡しを確認するアプリの作成

ページ・アイテムをパラメータとして渡して、別のページに遷移するために利用できる、いくつかの方法があります。

1. 動的アクションで実装する。
2. ボタンにターゲットを指定する。
3. 送信後にブランチする。

それぞれページ・アイテムの扱いが異なるので、アプリケーションを作ってみました。

このアプリケーションは、Oracle APEXの動作を理解するためのもので、作成方法自体は解説しません。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。インポートすると動作確認ができます。

https://github.com/ujnak/apexapps/blob/master/exports/pageitemvalues.sql

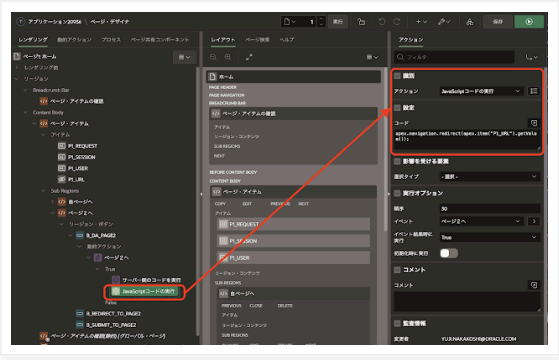
動的アクションによる遷移

動的アクションでページ遷移を行います。

ページ・アイテムのP1_REQUEST、P1_SESSION、P1_USERの値を遷移先のP2_REQUEST、P2_SESSION、P2_USERへ設定します。

単純なページ遷移であればURLを引数として、apex.navigation.redirectを呼び出すことでページの遷移が行われます。

```
apex.navigation.redirect(apex.item("P1_URL").getValue());
```

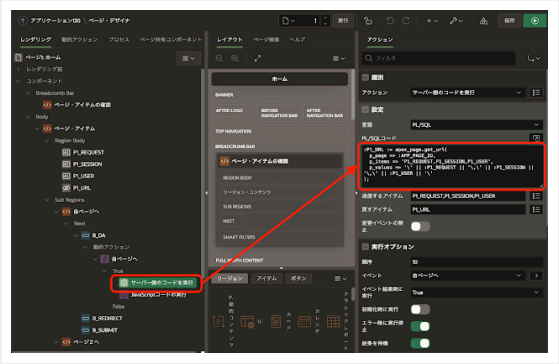


通常はページ遷移の際に引数としてチェックサムを追加する必要があります。これはサーバー側でしか計算できません。そのため、サーバー側でAPEX_PAGE.GET_URLを呼び出して、ページ遷移のためのURLを取得します。取得したURLは、ページ・アイテムP1_URLに戻します。

```
:P1_URL := apex_page.get_url(
  p_page => :APP_PAGE_ID,
  p_items => 'P1_REQUEST,P1_SESSION,P1_USER',
  p_values => '\' || :P1_REQUEST || '\,\' || :P1_SESSION || '\,\' || :P1_USER || '\'
);
```

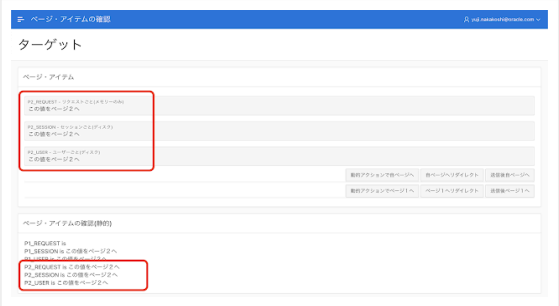
動的アクションで上記の計算を行うには、p_valuesとして渡される値を、送信するアイテムとして指定します。

ページ・アイテムにカンマ(,)やコロン(:)が含まれる可能性がある場合は、ページ・アイテムの値をバックスラッシュで囲みます。これにより、ページ・アイテムの中間で値が分割されることを防ぎます。



送信するアイテムとして指定されているので、URLはボタンをクリックした時点でのページ・アイテムの値が反映されます。よって、遷移先のページ・アイテムの値は、画面上の値になります。

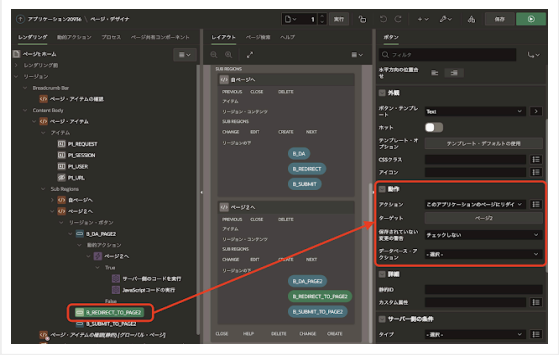
実際にボタンをクリックしてページ遷移をします。遷移先のページ・アイテムに画面上に入力されていた値が渡されていることが確認できます。



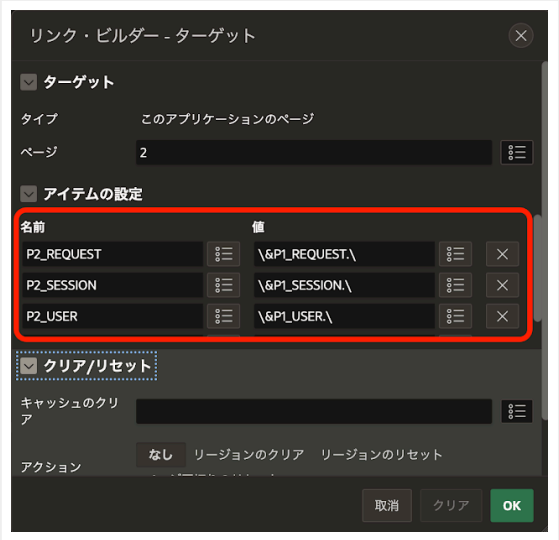
ボタンによるリダイレクト

最も一般的なページ遷移の方法だと思います。ページ・アイテムの値を引き継いで、別ページに遷移します。

ボタンの**動作**の**アクション**として、**このアプリケーションのページにリダイレクト**を選択し、**ターゲット**となるページを設定します。

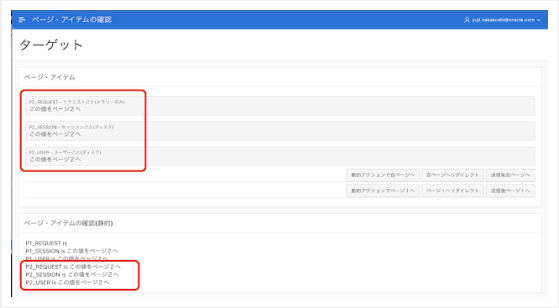


ターゲットの設定で、引き渡す**アイテムの設定**を行います。



名前として遷移先のページ・アイテムの名前、**値**として、このボタンが存在しているページのページ・アイテムの値を指定します。ここで値が&P1_REQUEST.といった置換文字列として指定されています。置換文字列の評価はページが生成されるときなので、引き渡される値はページが生成された時点での値になります。ページ生成後に画面に入力された値は渡されません。

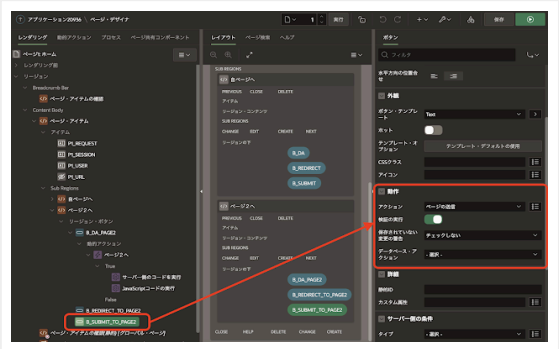
実際にページを遷移して確認できます。



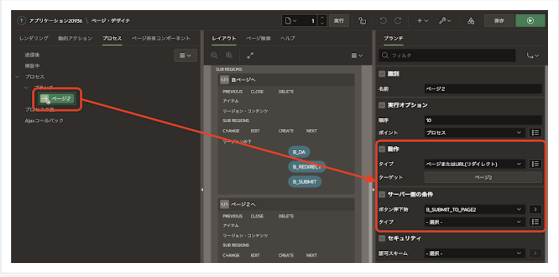
送信後にブランチ

通常はフォームを送信して、レポートのページに戻る、という処理で使われている設定です。ページを遷移させるために使用することはあまりありません。

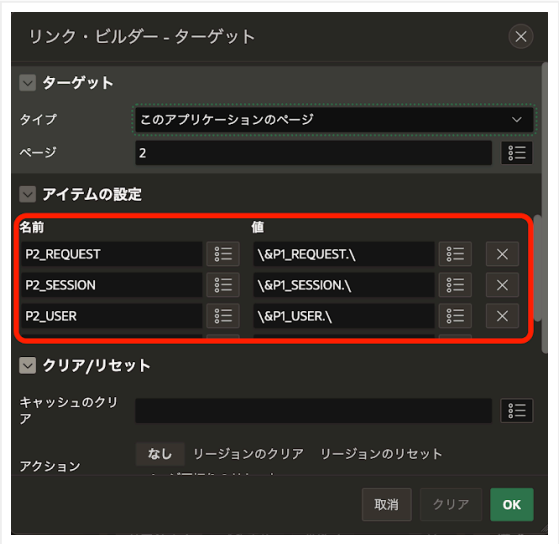
ボタンの**アクション**を**ページの送信**として、ブランチによってページ遷移をします。



ブランチの**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。ブランチがボタンを押したときのみに動作するように、**サーバー側の条件**として**ボタン押下時**にボタン名を指定します。



ターゲットの**アイテムの設定**は、リダイレクトの場合と同じです。



ボタンの**アクション**が**ページの送信**の場合、画面上のページ・アイテムの値はすべてHTTPのPOSTリクエストとしてサーバーへ送信されます。そのため、ここに現れている置換文字列はボタンを押した時点での画面上の値になります。

実際にページを遷移して確認できます。

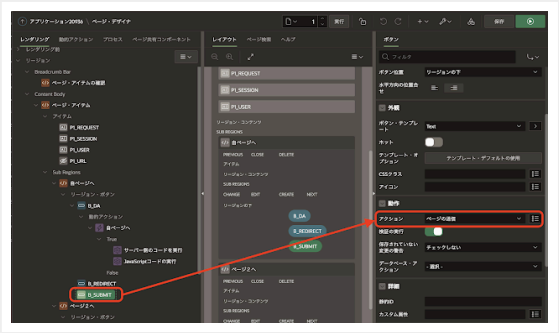
セッション・ステートの保持

セッション・ステートの保持の設定が異なる３つのページ・アイテムを作成しています。

- P1_REQUEST - リクエストごと(メモリーのみ)
- P1_SESSION - セッションごと(ディスク)
- P1_USER - ユーザーごと(ディスク)

ボタンのアクションが、単純にページの送信だけのときの動作を確認します。

この場合、単にページ・アイテムをサーバーに送信して、その後、自ページを再表示します。



それぞれのページ・アイテムに値を設定し、**送信後自ページへ**をクリックします。

セッション・ステートの保持が**リクエストごと(メモリーのみ)**と設定されているページ・アイテム**P1_REQUEST**の値は、ページの送信後、値が無くなっています。

P1_SESSIONおよびP1_USERは、サーバーが送信されたページ・アイテムを受け取った時点でディスクに保存されますが、リクエストごと(メモリーのみ)の場合は保存しません。送信されたページを処理するプロセス(プロセス・ビューに登録されている一連のプロセス)からは値を参照することが

できますが、それが終了し画面のレンダリングが実行される時点（レンダリング・ビューの処理）では、ページ・アイテムの値は消えています。

そう考えると、セッション・ステートはつねにディスクに保存した方がよい、と思えますが、以下の２点の理由より、基本は**リクエストごと(メモリーのみ)**を設定するのが望ましいです。

1. リクエストごと(メモリーのみ)はパフォーマンス面で有利です。
2. ディスクに保存は、グローバル変数に似ています。どのページからでも変更される可能性があるため、デバッグが難しくなります。メモリーのみの場合は、そのとき処理しているリクエスト内がページ・アイテムのスコープになります。

グローバルなスコープを持たせる場合は、ページ・アイテムではなくアプリケーション・アイテムの利用を検討すべきです。ページ・アイテムのソースをアプリケーション・アイテムとすることも可能です。

ページ・アイテムのセッション・ステートの保持をセッションごと(ディスク)、ユーザーごと(ディスク)として場合でも、異なるページのページ・アイテムを直接参照することは避けるべきです。ページをコピーしたり作り直した場合など、ページ番号は変わることがあるためです。ページを変更する際に、そのページ上のページ・アイテムを直接参照している他のページがあっても、それを発見するのは困難です。

１ページの中で処理を完結するようにすることにより、Oracle APEXのアプリケーションのデバッグが容易になります。

本記事は以上になります。

Oracle APEXのアプリケーション開発の一助になれば幸いです。

完

Yuji N. 時刻: 20:48

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