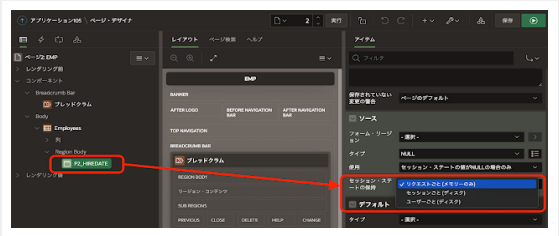
# 日日是Oracle APEX

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。

2022年8月10日水曜日

## セッション・ステートの保持 - メモリーとディスクの違い

ページ・アイテムの**ソース**の**セッション・ステートの保持**の設定で、**リクエストごと(メモリーのみ)**と**セッションごと(ディスク)**のどちらかを選んだ時の違いを、クラシック・レポートとポップアップLOVを使って説明してみます。

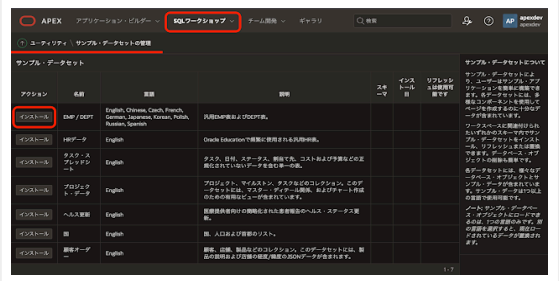


### 準備

**セッション・ステートの保持**の設定による動作の違いを確認するために、APEXアプリケーションを作成します。

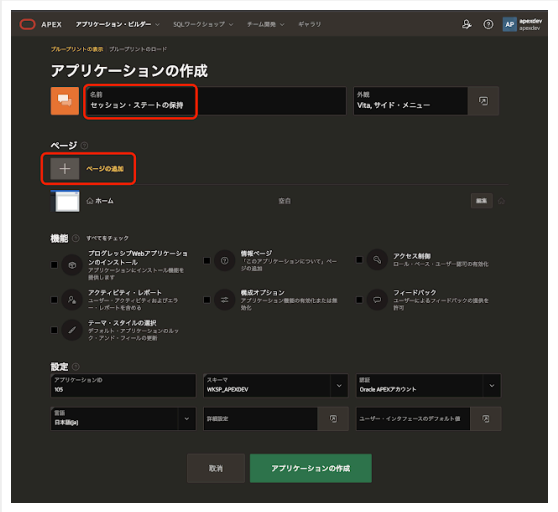
**サンプル・データセット**の**EMP/DEPT**に含まれる表**EMP**を検証に使用します。

**SQLワークショップ**の**ユーティリティ**の**サンプル・データセット**を開き、**EMP/DEPT**の**インストール**を実行します。**言語**は**日本語**と**英語**のどちらを選んでも作業は可能です。また、**アプリケーションの作成**は行いません。



**アプリケーション作成ウィザード**を実行します。

アプリケーションの**名前**を**セッション・ステートの保持**とし、**ページの作成**をクリックします。

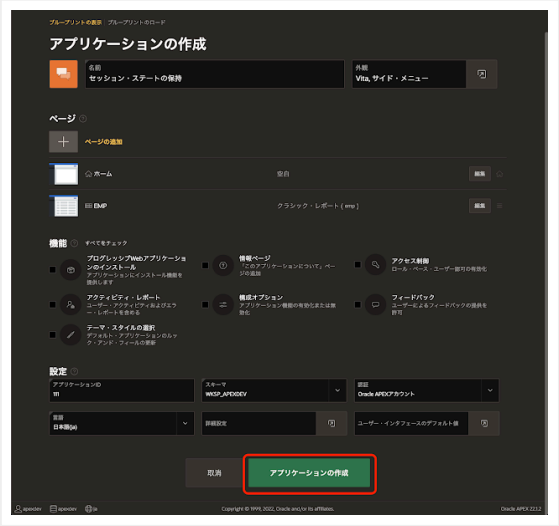



作成するのはクラシック・レポートですが、とりあえず**対話モード・レポート**を選択します。


**ページ名**は**EMP**とします。**表またはビュー**、**クラシック・レポート**を選択します。**表またはビュー**として**EMP**を選択します。使用するのはクラシック・レポートのみなので、**フォームを含める**の**チェックは入れません**。

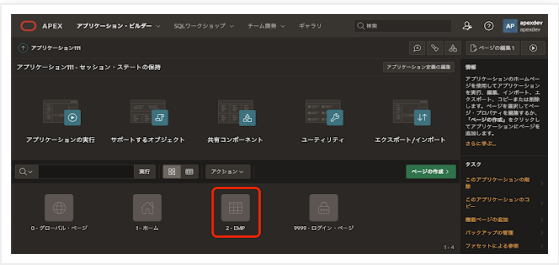
**ページの追加**をクリックします。


**アプリケーションの作成**を実行します。



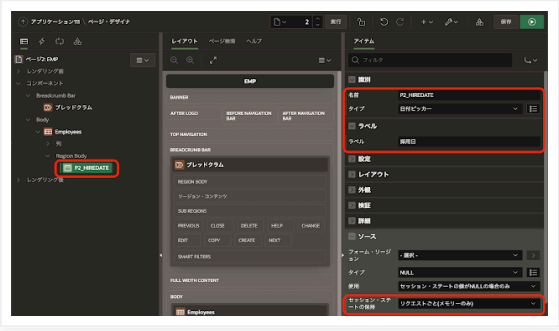

アプリケーションが作成されます。

**ページ・デザイナ**にてクラシック・レポートのページ（ページ番号**2**）を開きます。

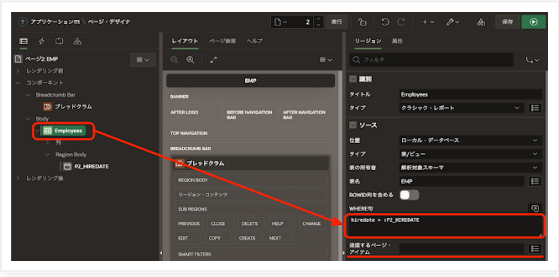



クラシック・レポートのリージョン**Employees**に、ページ・アイテム**P2_HIREDATE**を作成します。**識別**の**タイプ**として**日付ピッカー**を選択し、**ラベル**は**採用日**とします。

最初は**ソース**の**セッション・ステートの保持**は**リクエストごと(メモリーのみ)**とします。




リージョン**Employees**のソースの**WHERE句**として **hiredate > :P2_HIREDATE** を記述します。**送信するページ・アイテム**は指定しません。



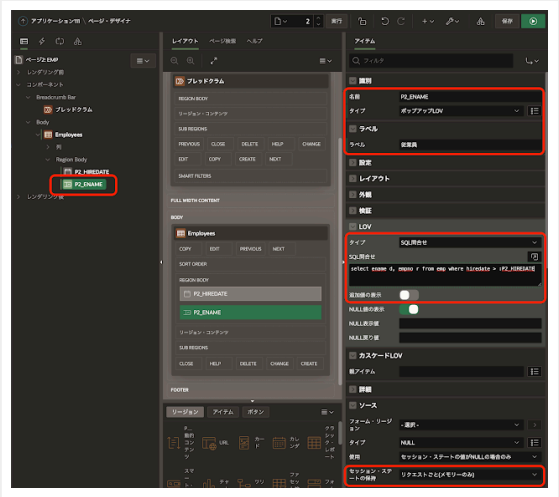

リージョンEmployeesに、ポップアップLOVのページ・アイテムを作成します。

**識別**の**名前**を**P2_ENAME**、**タイプ**を**ポップアップLOV**とします。**ラベル**は**従業員**とします。**LOV**の**タイプ**に**SQL問合せ**を選択し、**SQL問合せ**として、クラシック・レポートのソースと同等のSELECT文を記載します。

```
select ename d, empno r from emp where hiredate > :P2_HIREDATE
```

**追加値の表示**は**OFF**とします。

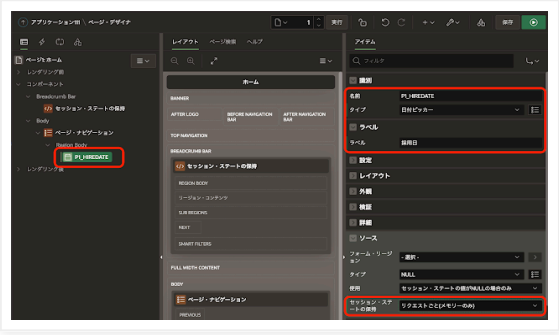
**ソース**の**セッション・ステートの保持**は**リクエストごと(メモリーのみ)**を選択します。今回のアプリケーションでは、ページ・アイテムP2_ENAMEの値が参照されることはないため、**セッション・ステートの保持**は**セッションごと(ディスク)**でもかまいません。どちらを設定しても良い場合は、サーバーへの負荷の少ない**リクエストごと(メモリーのみ)**を選びます。



このレポートを呼び出すボタンを、ホーム・ページに作成します。

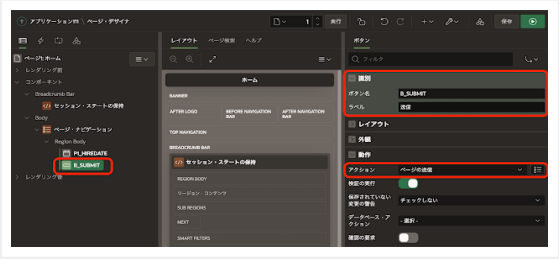
**ページ・デザイナ**で**ホーム・ページ**を開きます。

ページ・アイテム**P1_HIREDATE**を作成します。**識別**の**タイプ**は**日付ピッカー**、**ラベル**は**採用日**とします。**ソース**の**セッション・ステートの保持**は**リクエストごと(メモリーのみ)**を選択します。



クラシック・レポートのページに移動するボタンを作成します。

**識別**の**ボタン名**を**B_SUBMIT**、**ラベル**を**送信**とします。**動作**の**アクション**は**ページの送信**とします。



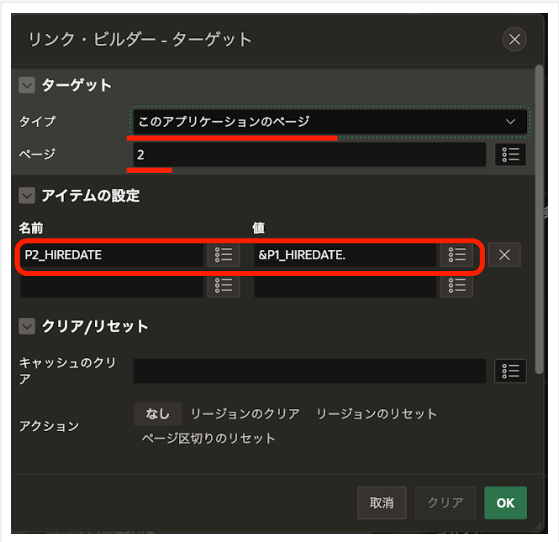
ページが送信された後にクラシック・レポートのページに遷移するために、ブランチを作成します。

左ペインで**プロセス・ビュー**を開きます。

**ブランチ**を作成し、**識別**の**名前**を**レポートを開く**とします。**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**B_SUBMIT**を選択します。

**リンク・ビルダー・ダーゲット**の設定です。

**ターゲット**の**タイプ**は**このアプリケーションのページ**、**ページ**は**2**になります。**アイテムの設定**の**名前**に**P2_HIREDATE**を選び、その**値**として**&P1_HIREDATE.**を指定します。




以上で検証に使用するアプリケーションは完成しました。

ここまでの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/maintain-session-state.zip

このアプリケーションを使って、検証を行います。

## クラシック・レポートへの遷移にブランチを使う点について

検証に使用するアプリケーションでは、ブランチを作成してクラシック・レポートのページに遷移しています。このときボタンの**動作**の**アクション**に**このアプリケーションのページにリダイレクト**を選択して、ページ番号２に遷移させると期待した動作にはなりません。

ボタンの**動作**の**アクション**に**このアプリケーションのページにリダイレクト**を設定すると、ボタンは**apex.navigation.redirect**を呼び出します。設定されたページ・アイテムはGETリクエストの引数として、サーバーに送信されます。GETリクエストの宛先となるURLはページの生成時に引数を含めて決定され、ページが表示された後のページ・アイテムの変更がURLに反映されることはありません。

簡略化すると、以下のHTML要素が生成されます。

**<button onclick='apex.navigation.redirect("/ords/r/apexdev/maintain-session-state/emp?p2_hiredate=レンダリング時のP1_HIREDATEの値&session=セッションID
");'>**

AタグのHREFの指定が、HTTPのGETで呼び出される動作と同じです。

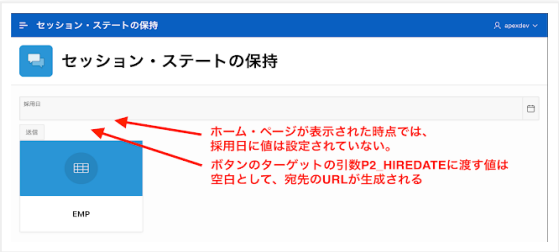
**動作**の**アクション**が**ページの送信**の場合は、**apex.submit**を呼び出します。画面上のページ・アイテムの値はPOSTリクエストの内容として、すべてサーバーに送信されます。

**<button onclick='apex.submit({request: "B_SUBMIT"; validate:true});'>**

formタグの内容が、HTTPのPOSTで送信される動作と同じです。

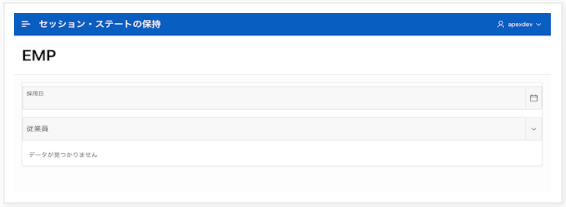
今回のボタン**B_SUBMIT**の**動作**の**アクション**を、**このアプリケーションのページにリダイレクト**に変更してページ番号２に遷移させてみます。ページ・アイテム**P1_HIREDATE**はホーム・ページが表示された時点では値が無いため、採用日に何を設定してもクラシック・レポートには従業員が表示されません。

ホーム・ページが表示されたときの状況です。




採用日に値を設定し、送信ボタンを押します。


送信ボタンのURLの引数P2_HIREDATEの値は空白なので、ターゲットのページに存在するクラシック・レポートには何も表示されません。

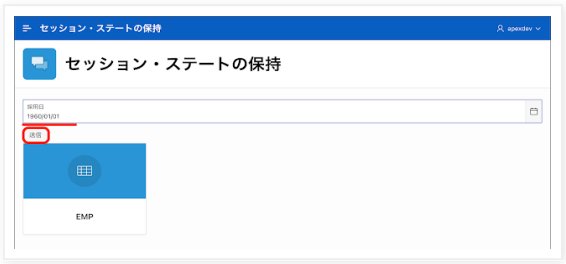



ボタンの**動作**の**アクション**として**ページの送信**が選択されていると、このような結果にはなりません。

## リクエストごと(メモリーのみ)の場合

ページ・アイテムP2_HIREDATEの**ソース**の**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)**のときの動作について確認してみます。

**採用日**に**1960/01/01**を入力し、**送信**をクリックします。

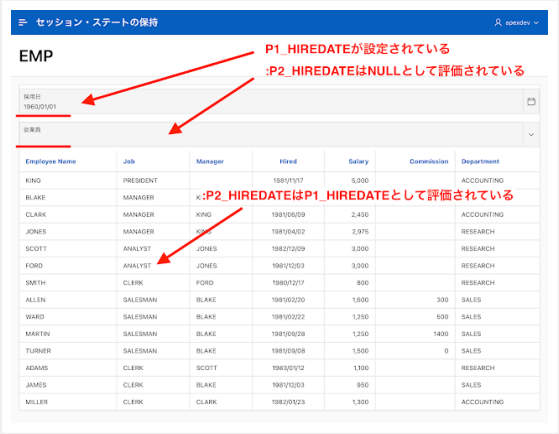



ページ・アイテム**P1_HIREDATE**の値**1960/01/01**が、P2_HIREDATEに渡されます。

ページ・アイテム**P2_HIREDATE**には**1960/01/01**が表示されます。

ポップアップLOV**P2_ENAME**の**SQL問合せ**に含まれる**P2_HIREDATEはNULL**となり、従業員は表示されません。

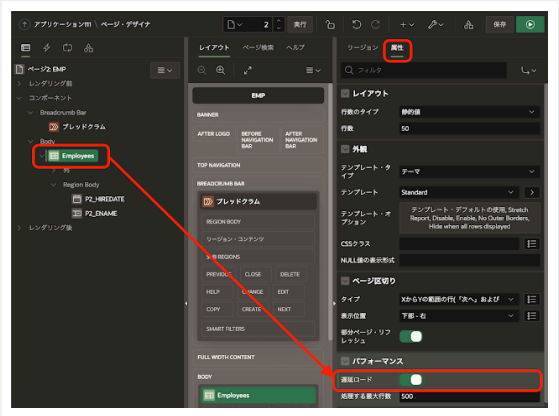
クラシック・レポートは**P2_HIREDATE**に**1960/01/01**が割り当てられ、**すべての従業員が一覧**されます。



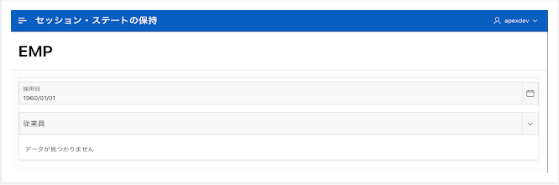

クラシック・レポートにはP2_HIREDATEに1960/01/01といった値が割り当てられるのに、ポップアップLOVはなぜNULLになるのか、その違いですが、コンポーネント自体がAjaxコールを発行してデータを取得している場合、バインド変数にページ・アイテムの値を割り当てるには、**送信するページ・アイテム**にそのページ・アイテムを設定する必要があります。

動的アクションのリフレッシュに対応しているコンポーネントは、概ねコンポーネントの表示（HTMLの生成）とデータの取得は独立しているので、データ・ソースにバインド変数が含まれる場合は**送信するページ・アイテム**の設定が必要になります。

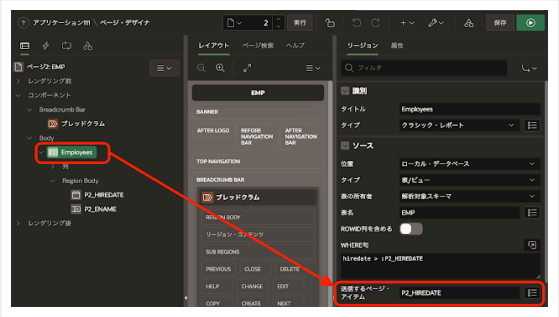
上記の設定ではクラシック・レポートで従業員の一覧が表示されています。クラシック・レポートの**属性**の**パフォーマンス**の**遅延ロード**を**ON**にすると、レポートの表示（HTMLの生成）とデータの取得が非同期で行われるようになります。



この場合、**送信するページ・アイテム**を設定していないと、**P2_HIREDATE**が**NULL**になり従業員が表示されません。

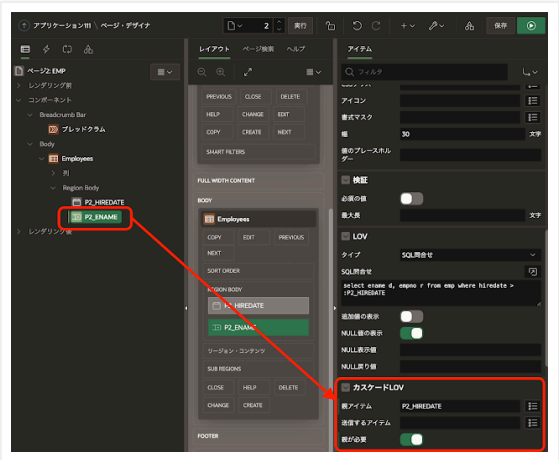



ソースの**送信するページ・アイテム**に**P2_HIREDATE**を設定すると、ソースのSELECT文を実行する際にはページ・アイテム**P2_HIREDATE**の画面上の値を取得し、サーバーに送信します。そのため、取得される従業員はつねにページ・アイテムP2_HIREDATEが評価された結果になります。

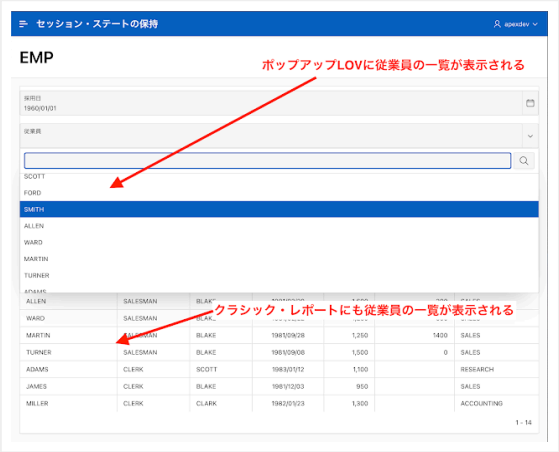


ポップアップLOVでは**カスケードLOV**の設定があり、ここで設定したページ・アイテムは、**LOV**の**SQL問合せ**が実行されるときにサーバーに送信されます。

ページ・アイテム**P2_ENAME**の**カスケードLOV**の**親アイテム**として**P2_HIREDATE**を設定することにより、ポップアップLOVでも従業員の一覧が表示されます。



これらの設定を行うことにより、クラシック・レポートおよびポップアップLOVの双方で、データ取得時にページ・アイテム**P2_HIREDATE**が評価されます。

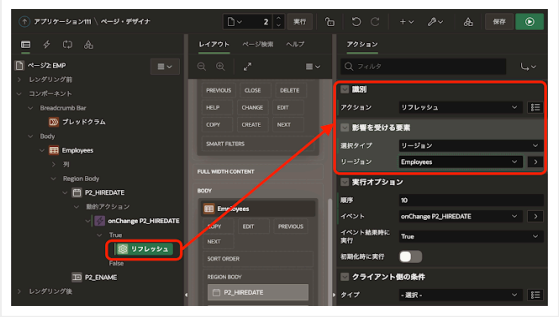



多くのコンポーネントは**送信するページ・アイテム**の設定を含んでいます。送信するページ・アイテムによる設定では、その時点で画面に表示されている値がサーバーに渡されます。

ページ・アイテムP2_HIREDATEが変更されたときにクラシック・レポートがリフレッシュされるよう、動的アクションを作成します。

ページ・アイテム**P2_HIREDATE**に**動的アクションを作成**します。**名前**を**onChange P2_HIREDATE**とします。**タイミング**は**デフォルト**がそのまま使えます。**TRUEアクション**に**リフレッシュ**、**影響を**

**受ける要素**の**選択タイプ**を**リージョン**、**リージョン**として**Employees**を選択します。



採用日の値が変更されると、クラシック・レポートの**リフレッシュ**が呼び出されます。データ・ソースとなるSELECT文が実行されるときはつねに**送信するページ・アイテム**として指定されているページ・アイテムの値もサーバーに送信されるため、変更された採用日が**P2_HIREDATE**に割り当てられた結果が、従業員の一覧として表示されます。

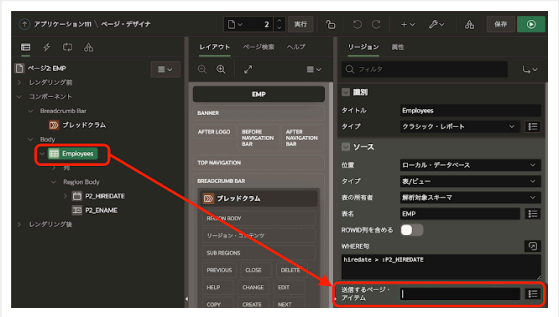
ポップアップLOVは**カスケードLOV**の**親アイテム**として**P2_HIREDATE**が設定されているため、クラシック・レポートと同様に変更された採用日でLOVの一覧に置き換えられます。

## セッションごと(ディスク)の場合

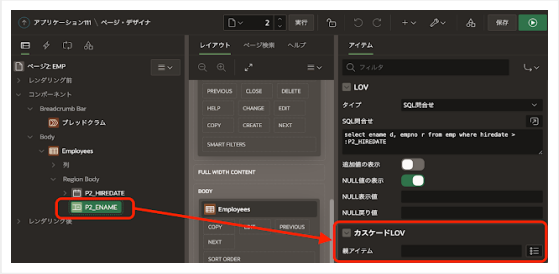
ページ・アイテム**P2_HIREDATE**の**ソース**の**セッション・ステートの保持**を**セッションごと(ディスク)**に変更して、動作を確認します。

今までの設定を残したままだと、ほとんど動作に違いはありません。**送信するページ・アイテム**が設定されていると、データ・ソースを評価する際に**セッション・ステートの保持**がどちらであっても、画面上の値をサーバーに送信するためです。

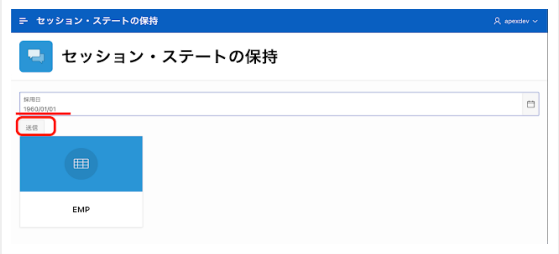
そのため、クラシック・レポートから**送信するページ・アイテム**の設定を外します。**遅延ロード**は**ON**のまま変更しません。



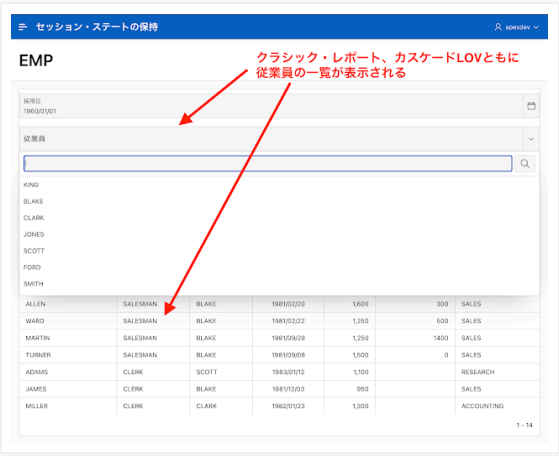
また、**カスケードLOV**からは**親アイテム**の設定を外します。



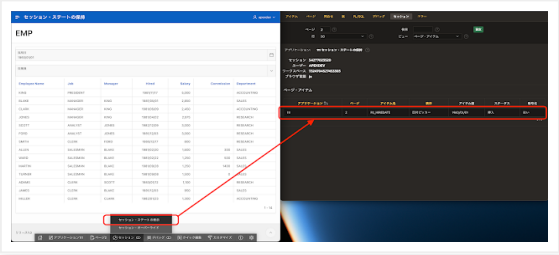
**採用日**に**1960/01/01**を入力し、**送信**します。




**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)**のときとは異なり、クラシック・レポート、ポップアップLOVともに、ページ・アイテム**P2_HIREDATE**が**1960/01/01**を割り当てた検索結果が、従業員として一覧されます。**セッション・ステートの保持**が**セッションごと(ディスク)**となっているため、ページ・アイテム**P2_HIREDATE**に**1960/01/01**が設定された時点で、セッション・ステートとしてデータベースに保持されています。データ・ソースにバインド変数としてP2_HIREDATEが使われている場合、セッション・ステートに保持されているP2_HIREDATEの値が割り当てられます。




セッション・ステートに保持されているアイテムの値は、**開発者ツール・バー**の**セッション**の**セッション・ステートの表示**から参照することができます。この画面から参照できる値が、バインド変数P2_HIREDATEに割り当てられています。

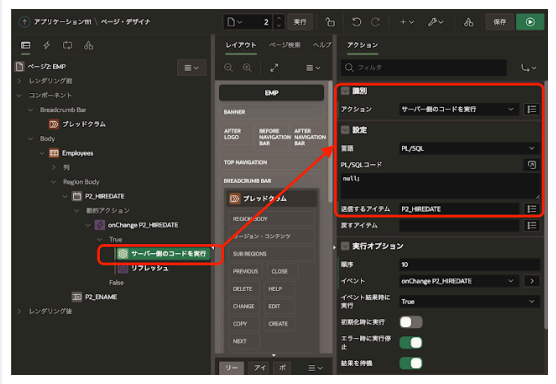


この状態で、採用日を2022/08/01に変更します。採用日が2022/08/01以降の従業員は存在しません。動的アクションは有効なので、リフレッシュされたクラシック・レポートには従業員は表示されないことが期待されます。しかし、実際はそうならず、従業員の一覧は変わりません。

採用日の変更は画面上だけで、サーバー側に送信されていないためです。

変更されたページ・アイテムの値をサーバーに送信するため、リフレッシュの直前にTRUEアクションを作成します。

**識別**の**アクション**として**サーバー側のコードを実行**を選択し、**設定**の**PL/SQLコード**として**null;**を記述します。**送信するアイテム**として**P2_HIREDATE**を指定します。コードには何も書いていないので、単にP2_HIREDATEの値をサーバーに送信するアクションです。

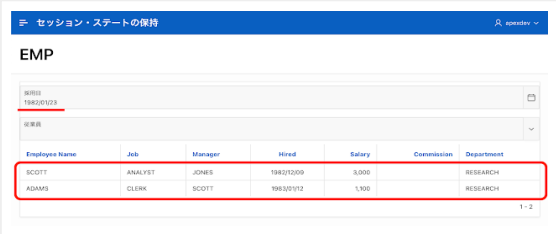


ページ・アイテムの値が送信されると、セッション・ステートにその値が保存されます。クラシック・レポートやポップアップLOVで、リフレッシュが行われる場合は更新されたセッション・ステートがバインド変数に割り当てられます。

コンポーネントが送信するページ・アイテムの設定を持っているのであれば、わざわざ動的アクションを作成する必要は無いでしょう。

ホーム・ページより採用日に1982/01/23を指定して、クラシック・レポートのページを開きます。

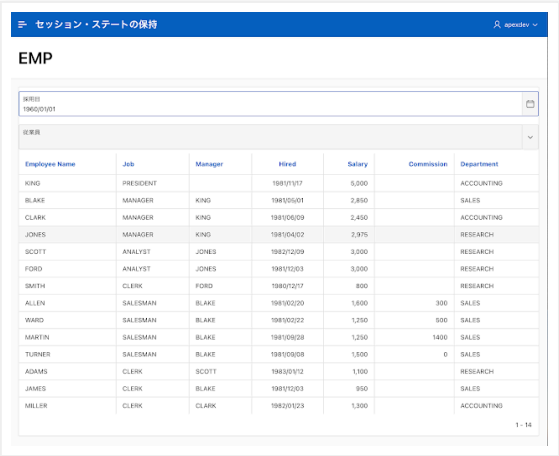
クラシック・レポートでは２名の従業員が選択されています。



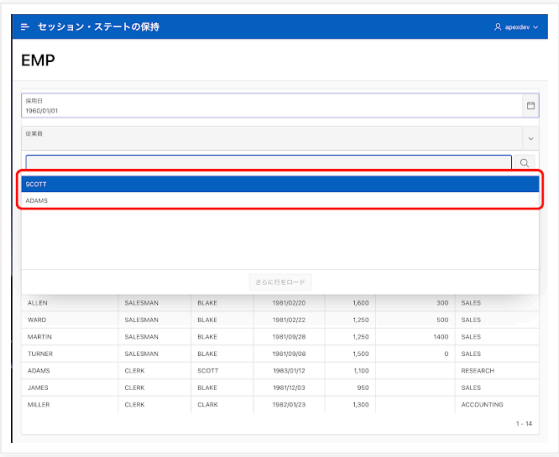

ポップアップLOVもSELECT文は同等なので、従業員は２名になります。


採用日を1960/01/01に変更します。動的アクションにより、ページ・アイテムP2_HIREDATEの値1960/01/01がサーバーに送信され、クラシック・レポートがリフレッシュされます。

結果としてクラシック・レポートにはすべての従業員が一覧されます。




ポップアップLOVには変更がなく、従業員は２名です。ポップアップLOVはリフレッシュされていないためです。ポップアップLOVをリフレッシュするためには、**カスケードLOVの親アイテム**の設定が必要です。




## まとめ

1. 画面上の値をSQLのバインド変数に割り当てるには、**送信するページ・アイテム**の設定を使用する。もしくは、それと同等の設定を使用する。(ポップアップLOVの場合は**カスケードLOV**)。

2. **送信するページ・アイテム**にてバインド変数にページ・アイテムの値を割り当てている場合、セッション・ステートに値を保持する必要性はほぼない。
3. **送信するページ・アイテム**の設定を持たないが**リフレッシュ**が可能なコンポーネントであれば、**セッション・ステートの保持**を**セッションごと(ディスク)**にし、動的アクションでページ・アイテムの値をサーバーに送信することで対応できる。
4. **送信するページ・アイテム**の設定がなく、**リフレッシュ**にも対応していないコンポーネントを更新するには、**ページの送信**を実施する必要がある。

以上です。

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: 14:52

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