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。

2024年6月6日木曜日

対話グリッドのセルへのマウス・エンターとマウス・リーブを確認する

対話グリッドのセルに対して、マウス・エンターとマウス・リーブのイベントで呼び出される動的アクションを作成したところ、当初考えていたように動きませんでした。

動作を確認するためのAPEXアプリケーションを作成し、**対話グリッド**の**セル**を**ターゲット**とした**マウス・エンター**と**マウス・リーブ**の**イベント**で呼び出される**動的アクション**を作成します。

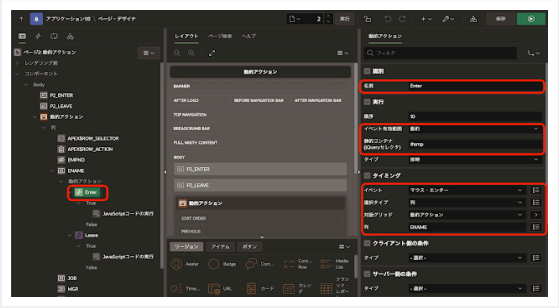
動作の確認には、**サンプル・データセット**のEMP/DEPTに含まれる表**EMP**をソースとした、編集可能対話グリッドを使用します。動的アクションは列**ENAME**に作成します。対話グリッドのリージョンに、**静的ID**として**emp**を設定します。

最初は、素直に対話グリッドの**列**に動的アクションを作成します。

識別の**名前**は**Enter**とします。**実行**の**イベント有効範囲**は**動的**、**静的コンテナ(jQueryセレクタ)**は**#emp**とします。対話グリッドのセルはページのリロードをせずに書き換わるため、**イベント有効範囲**を**動的**にしています。

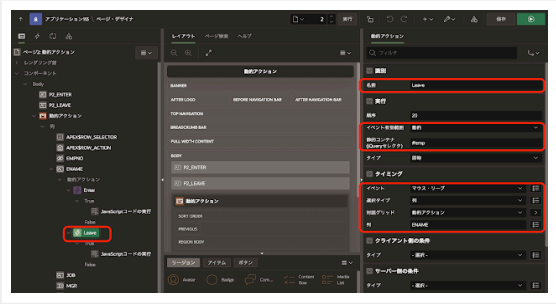
タイミングの**イベント**は**マウス・エンター**、**選択タイプ**は**列**、**列**は**ENAME**を指定します。**マウス・エンター**の**イベント**が発生したときに実行される**JavaScriptコード**は以下です。

```
console.log(this.triggeringElement);
apex.items.P2_ENTER.setValue($v(this.triggeringElement));
```



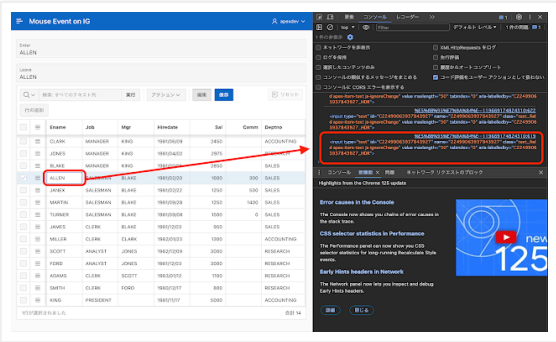
マウス・リーブのイベントで呼び出される動的アクションも作成します。**イベント**が**マウス・リーブ**になるのと、実行される**JavaScriptコード**が異なります。

```
console.log(this.triggeringElement);
apex.items.P2_LEAVE.setValue($v(this.triggeringElement));
```



ページを実行してマウス・エンターおよびマウス・リーブのイベントの発生を確認すると、イベントのトリガー要素（**this.triggeringElement**）が、INPUT要素であることが確認できます。

<input type="text" id="C22499063937843927" name="22499063937843927" class="text_field apex-item-text js-ignoreChange" value="" maxlength="50" tabindex="0" aria-labelledby="C22499063937843927_HDR">

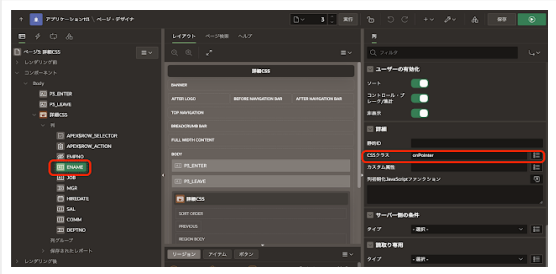


対話グリッドのセルがINPUT要素になるのは、セルをダブルクリックなどして編集する状態になったときです。セルが表示されているときはINPUT要素ではないため、その状態のセルにポインタを載せてもイベントは発生しません。

対話グリッドの列に動的アクションを作成する場合、**タイミング**の**選択タイプ**、**対話グリッド**、**列**の３つを指定します。この形式で指定すると、編集モードになったセルがトリガー要素になることがわかりました。

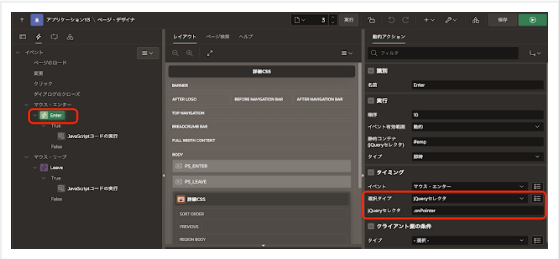
編集モードになったセルではなく、表示されているセルがトリガー要素になる設定を探してみます。

列**ENAME**の**詳細**の**CSSクラス**に、擬似CSSクラスとして**onPointer**を設定します。



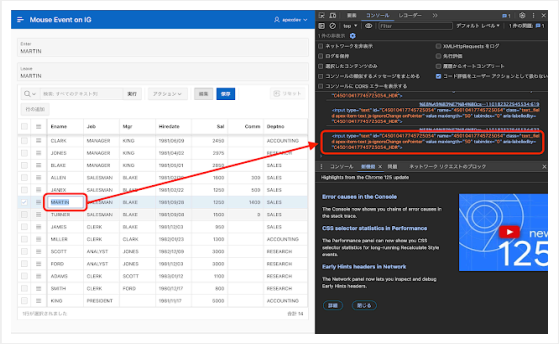
マウス・エンター/リーブの**タイミング**の**選択タイプ**を**jQueryセレクタ**に変更し、**jQueryセレクタ**として**.onPointer**を指定します。

列ENAMEの一番内部の要素がトリガー要素になります。

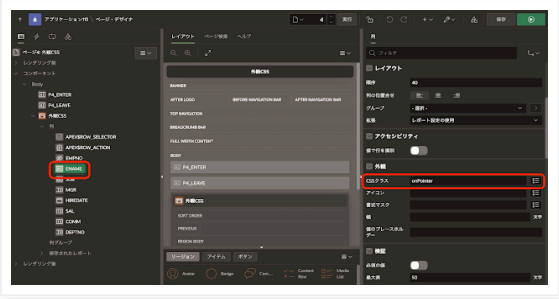


ページを実行してマウス・エンターおよびマウス・リーブのイベントの発生を確認します。

先ほどの列ENAMEに動的アクションを作成したときと同じ動作が確認できます。対話グリッドの**列**の**詳細**の**CSSクラス**は、INPUT要素に設定されています。そのため、この設定では表示中のセルをターゲットとしたマウス・イベントは取れません。



列**ENAME**の**詳細**の**CSSクラス**として設定した**onPointer**を削除し、代わりに**外観**の**CSSクラス**として設定します。



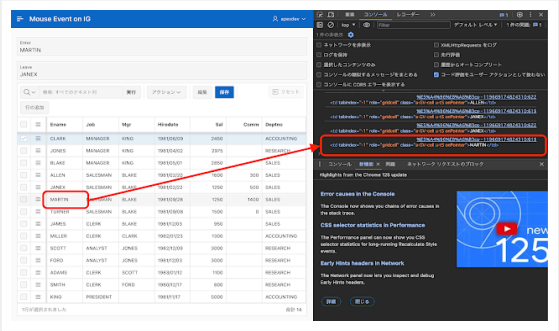
動的アクションのタイミングはCSSクラスonPointerに設定されているため変更は不要ですが、実行するJavaScriptは**$v(this.triggeringElement)**の代わりに**this.triggeringElement.textContent**をページ・アイテムに設定するように変更します。

```
console.log(this.triggeringElement);
apex.items.P4_ENTER.setValue(this.triggeringElement.textContent);
```

ページを実行してマウス・エンターおよびマウス・リーブのイベントの発生を確認すると、イベントのトリガー要素（**this.triggeringElement**）が、TD要素であることが確認できます。

<td tabindex="-1" role="gridcell" class="a-GV-cell u-tS onPointer">MARTIN</td>

この場合はセルが編集モードになる必要はなく、セルの上にマウスが載ったらマウス・エンター、離れたらマウス・リーブのイベントが発生します。

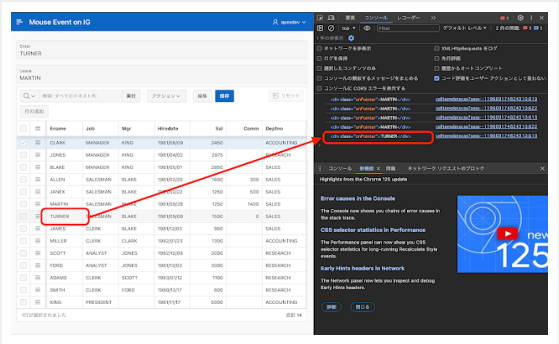


外観の**CSSクラス**の代わりに列**ENAME**に**cellTemplate**を設定します。

```
function(options) {
   options.defaultGridColumnOptions = {
      cellTemplate: '<div class="onPointer">&ENAME.</div>'
   }
   return options;
}
```

この場合、cellTemplateとして設定した要素がトリガー要素になります。セルが編集モードになる必要はありません。

** TURNER **



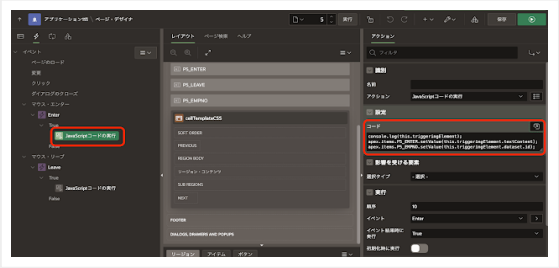
外観の**CSSクラス**として擬似クラスonPointerを設定しても、**cellTemplate**のclass指定にonPointerを含めても、表示モードのセル上でマウス・エンター/リーブのイベントを取ることができます。

cellTemplateを使うとカスタム属性などを追加できるため、呼び出されるJavaScriptの記述が楽になるという利点があります。例えばcellTemplateにカスタム属性data-idとして従業員番号を渡します。

```
function(options) {
   options.defaultGridColumnOptions = {
      cellTemplate: '<div data-id=&EMPNO. class="onPointer">&ENAME.</div>'
   }
   return options;
}
```

動的アクションで呼び出されるJavaScriptでは、this.triggeringElement.dataset.idとして従業員番号を参照できます。

```
console.log(this.triggeringElement);
apex.items.P5_ENTER.setValue(this.triggeringElement.textContent);
apex.items.P5_EMPNO.setValue(this.triggeringElement.dataset.id);
```



今回の記事は以上です。

今回の作業で使用した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/mouse-event-on-ig.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: 18:15

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