日日是Oracle APEX

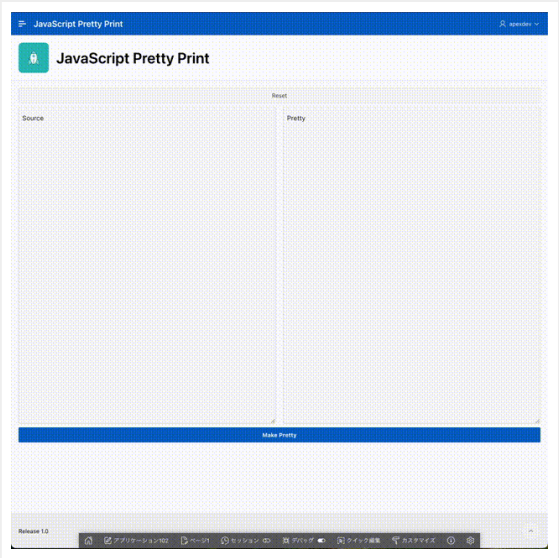
Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2024年6月11日火曜日

MLEで動作するJavaScriptへ渡す文字列がVARCHAR2とCLOBで扱いが異なる

Oracle Database 23aiのMLEを使って、JavaScriptの整形を行うサンプル・アプリケーションを作ってみました。整形には、こちらの記事「NPMパッケージからOracle Datase 23aiのMLEモジュールを作成する」でMLEモジュールとして作成したpretty-jsを使っています。

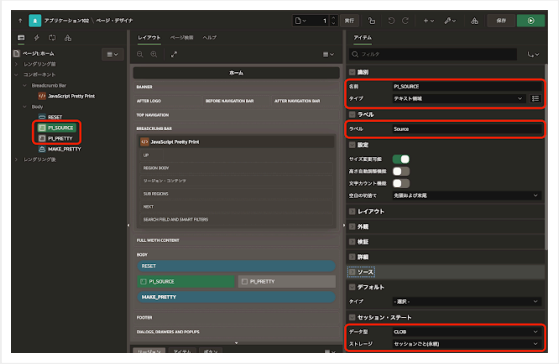
以下のように動作します。



APEXアプリケーションのエクスポートを以下に置いています。
https://github.com/ujnak/apexapps/blob/master/exports/mle-javascript-pretty-print.zip

アプリケーションでは、ボタン**MAKE_PRETTY**のクリックで、ページ・アイテム**P1_SOURCE**に入力した文字列をMLEモジュールの**prettyJs**を呼び出して整形し、結果をページ・アイテム**P1_PRETTY**に設定しています。

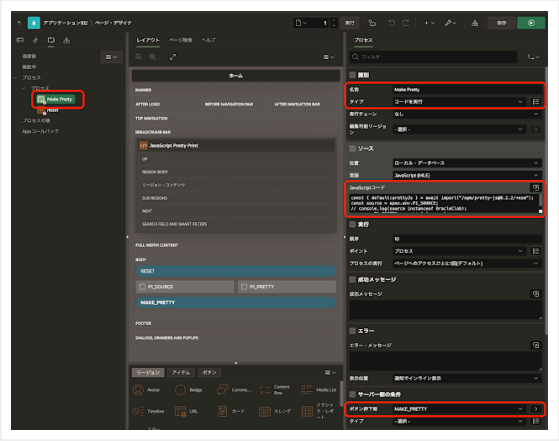
セッション・ステートの**データ型**に**CLOB**を指定しています。



ボタンMAKE_PRETTYをクリックしたときに実行するJavaScriptコードです。

```
const { default:prettyJs } = await import("/npm/pretty-js@0.2.2/+esm");
const source = apex.env.P1_SOURCE;
// console.log(source instanceof OracleClob);
apex.env.P1_PRETTY = prettyJs(source);
```

すでにMLEモジュールができていることもありますが、特に文字列を扱う処理は、JavaScriptの方がPL/SQLよりも容易に実装できるように思います。



JavaScriptを整形するアプリケーションは簡単にできあがりました。

これから本題ですが、JavaScriptのファンクション**prettyJs**は**引数**として**文字列**を受け取り、**戻り値**として**文字列**を返します。

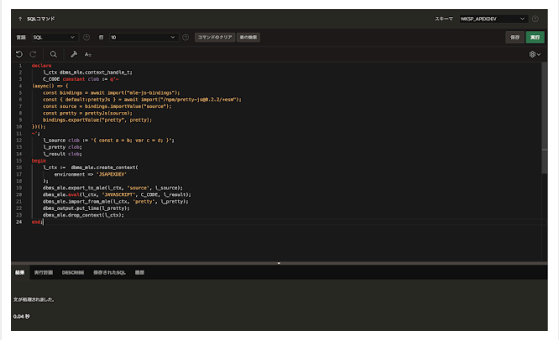
APEXのプロセスとしてJavaScriptコードを実行する場合は、DBMS_MLE.EVALの呼び出しに必要なMLEコンテキストの準備、DB接続およびページ・アイテムへの値の受け渡しなどはAPEXが行います。

APEXのプロセスとして記述したJavaScriptコードを**SQLコマンド**で実行するためには、以下のようなコードを記述します。

```
declare
   l_ctx dbms_mle.context_handle_t;
   C_CODE constant clob := q'~
(async() => {
   const bindings = await import("mle-js-bindings");
   const { default:prettyJs } = await import("/npm/pretty-js@0.2.2/+esm");
   const source = bindings.importValue("source");
   const pretty = prettyJs(source);
   bindings.exportValue("pretty", pretty);
})();
```

```
~';
  l_source clob := '{ const a = b; var c = d; }';
  l_pretty clob;
  l_result clob;
begin
  l_ctx :=  dbms_mle.create_context(
    environment => 'JSAPEXDEV'
  );
  dbms_mle.export_to_mle(l_ctx, 'source', l_source);
  dbms_mle.eval(l_ctx, 'JAVASCRIPT', C_CODE, l_result);
  dbms_mle.import_from_mle(l_ctx, 'pretty', l_pretty);
  dbms_output.put_line(l_pretty);
  dbms_mle.drop_context(l_ctx);
end;
```

しかし、上記のコードではJavaScriptファンクションのprettyJsの戻り値は空文字になります。



以下のコードで変数l_sourceをMLEへエクスポートしています。

```
dbms_mle.export_to_mle(l_ctx, 'source', l_source);
```

ここで、l_sourceはclobとして定義されています。

```
l_source clob := '{ const a = b; var c = d; }';
```

この場合、MLE側でインポートした値は**OracleClob**のインスタンスとなります。

```
const source = bindings.importValue("source");
```

上記のコードは、第２引数を省略せずに記述すると次のようになります。

```
const source = bindings.importValue("source", bindings.JSTypes.ORACLE_CLOB);
```

OracleClobが持つプロパティやメソッドは、以下のページで紹介されています。

https://oracle-samples.github.io/mle-modules/docs/mle-js-plsqltypes/23ai/classes/IOracleClob.html

Oracle Database 23aiで使用できるMLEモジュールの説明は、以下のページにまとまっています。

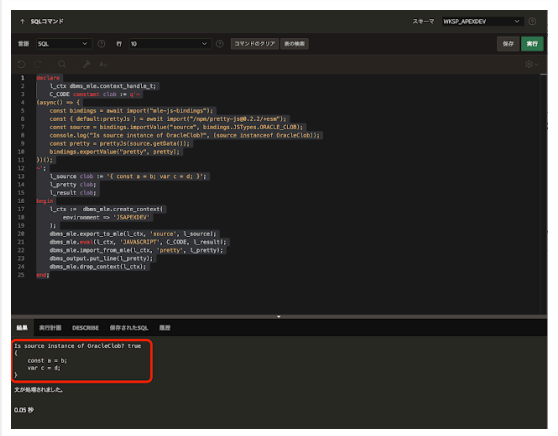
https://oracle-samples.github.io/mle-modules/

PL/SQL側よりCLOBが渡されたときはMLE側では**OracleClob**として取り出されるため、getData()を呼び出して文字列を取り出す必要があります。

以上を考慮して最初のコードを書き直します。

```
declare
   l_ctx dbms_mle.context_handle_t;
   C_CODE constant clob := q'~
(async() => {
   const bindings = await import("mle-js-bindings");
   const { default:prettyJs } = await import("/npm/pretty-js@0.2.2/+esm");
   const source = bindings.importValue("source", bindings.JSTypes.ORACLE_CLOB);
   console.log("Is source instance of OracleClob?", (source instanceof OracleClob));
   const pretty = prettyJs(source.getData());
   bindings.exportValue("pretty", pretty);
})();
~';
   l_source clob := '{ const a = b; var c = d; }';
   l_pretty clob;
   l_result clob;
begin
   l_ctx :=  dbms_mle.create_context(
      environment => 'JSAPEXDEV'
   );
   dbms_mle.export_to_mle(l_ctx, 'source', l_source);
   dbms_mle.eval(l_ctx, 'JAVASCRIPT', C_CODE, l_result);
   dbms_mle.import_from_mle(l_ctx, 'pretty', l_pretty);
   dbms_output.put_line(l_pretty);
   dbms_mle.drop_context(l_ctx);
end;
```

今度は整形されたJavaScriptのコードが出力されます。



MLEモジュールmle-js-plsqltypesとして、OracleClob以外にもPL/SQLの型に対応したクラスが定義されています。

https://oracle-samples.github.io/mle-modules/docs/mle-js-plsqltypes/23ai/

APEXのページ・アイテムは型がなく、基本的に文字列です。コード中でページ・アイテムを日付や数値として扱う際には、暗黙の型変換が行われます。それはapex.envから参照されるページ・アイテムについても同様で、値は文字列として保持されています。（恐らく）MLE JavaScriptでは型変換は自動的に行われないので、MLEとPL/SQLで値をやり取りする場合は、常に変数の型を意識する必要があるでしょう。

今回の記事は以上になります。

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: 12:07

共有

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