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4月21日水曜日

Excelファイルのアップロード - その１- 表APEX_APPLICATION_TEMP_FILESを使う方法

以下のようなExcelファイルのアップロードを行う実装を行ってみます。Excelファイルのファイル名はcitylist.xlsxとします。

PREFECTURE	CITY	COUNT
北海道	札幌市	100
宮城県	仙台市	200
東京都	中央区	1000
新潟県	新潟市	400
大阪府	大阪市	100
広島県	広島市	300
高知県	高知市	90
福岡県	福岡市	800

その１として、
表APEX_APPLICATION_TEMP_FILESを使う方法
その２として、
作成済みの表のBLOB列を使う方法
の２つの方法に取り組みます。

最初であるこの記事では、表APEX_APPLICATION_TEMP_FILESを使った方法を実装します。

ファイルのアップロードを行うアプリケーションは、いままでに何件か記事を書いています。今回は、Oracle APEX 21.1で実装される予定のデータ・ロードの機能を手作業で作ることで、その実装を理解することを目的としています。

アップロードする表は、以下のクイックSQLのモデルによって作成します。

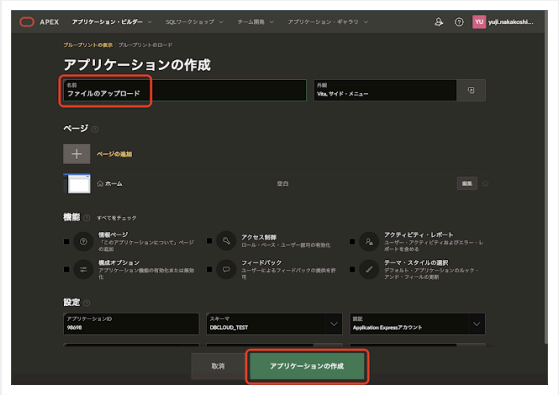
```
# prefix: fup
# semantics: default
citylist
    prefecture vc80
    city vc80
    count num
```

実際に実行するDDLとしては以下になります。

```
create table fup_citylist (
    id                          number generated by default on null as identity
                                constraint fup_citylist_id_pk primary key,
    prefecture                  varchar2(80),
```

```
    city                        varchar2(80),
    count                       number
)
;
```

ファイルのアップロードを実装するために、空のアプリケーションを作成します。アプリケーションの**名前**は**ファイルのアップロード**とします。**アプリケーションの作成**を実行します。



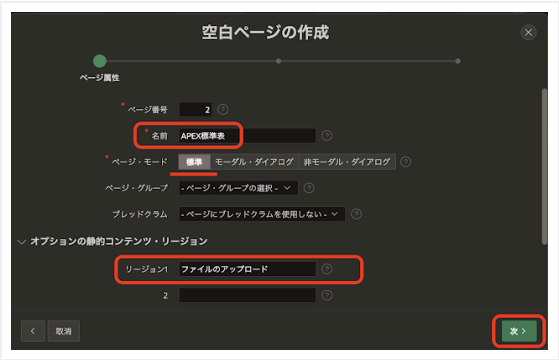
空のアプリケーションが作成されます。

最初にOracle APEXが提供している表APEX_APPLICATION_TEMP_FILESに一旦ファイルをアップロードしたのち、対象の表FUP_CITYLISTへデータの投入を行うページを作成します。

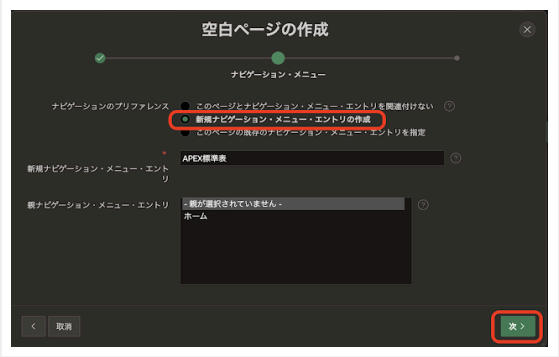
静的コンテンツのリージョンを含むページを作成します。**ページの作成**を実行し、ページ作成ウィザードを開始します。

空白ページをクリックします。

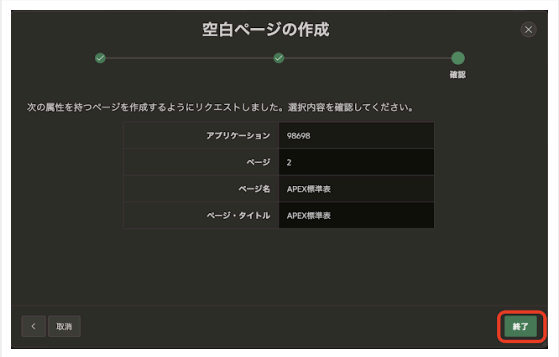
名前を**APEX標準表**、**ページ・モード**は**標準**、**オプションの静的コンテンツ・リージョン**として、リージョンをひとつ、**リージョン1**として**ファイルのアップロード**を作成します。**次**に進みます。



ナビゲーションのプリファレンスとして、**新規ナビゲーション・メニュー・エントリの作成**を選択します。**次**に進みます。

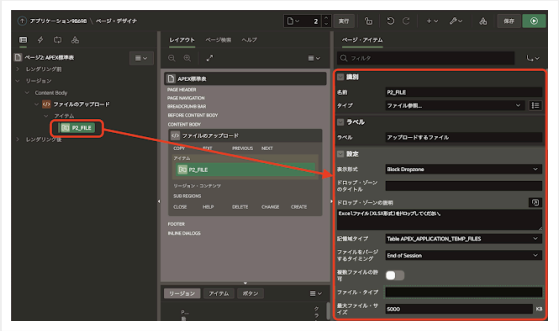


終了をクリックすると、静的コンテンツのリージョンがひとつあるページが作成されます。

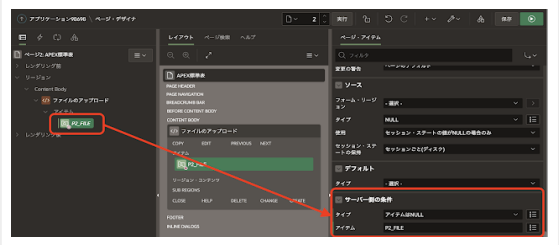


ページが作成されたら、アップロードするファイルを選択するページ・アイテムを作成します。**識別**の**名前**を**P2_FILE**、**タイプ**は**ファイル参照...**を指定します。**ラベル**として**アップロードするファイル**、**表示形式**として今回は**Block Dropzone**にしてみます。表示形式だけなので、どれを選んでも

ファイルは選択できます。**ドロップ・ゾーンの説明**として**Excelファイル(XLSX形式)をドロップしてください。**と記述します。**記憶域タイプ**として**Table APEX_APPLICATION_TEMP_FILES**を選択します。**ファイルをパージするタイミング**は**End of Session**とします。**最大ファイル・サイズ**として**5000KB**の制限を与えています。

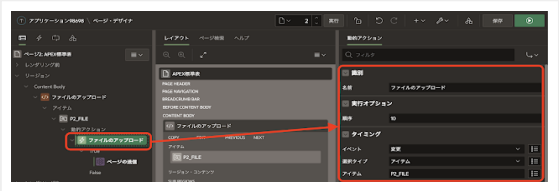


一旦ファイルが選択されたら、その後にページ・アイテムが変更されることを防ぐため、**サーバー側の条件**で**タイプ**に**アイテムがNULL**を選択し、**アイテム**に**P2_FILE**を指定します。この設定によりファイルが未選択のときに限り、ファイル参照が表示されます。

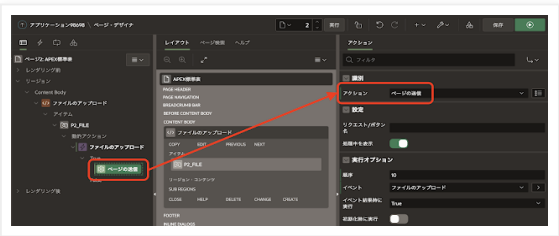


ファイルが選択された時点で表APEX_APPLICATION_TEMP_FILESに保存されるよう、動的アクションを作成します。ページ・アイテムP2_FILE上でコンテキスト・メニューを表示させ、**動的アクションの作成**を実行します。

動的アクションの**名前**を**ファイルのアップロード**とします。**タイミング**はデフォルトで**イベント**が**変更**、**選択タイプ**は**アイテム**、**アイテム**は**P2_FILE**となり、P2_FILEが変更されるとアクションが実行されます。



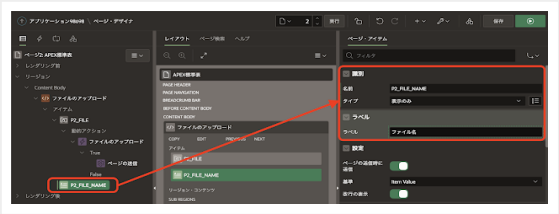
Trueアクションを選択し、**アクション**として**ページの送信**を選択します。



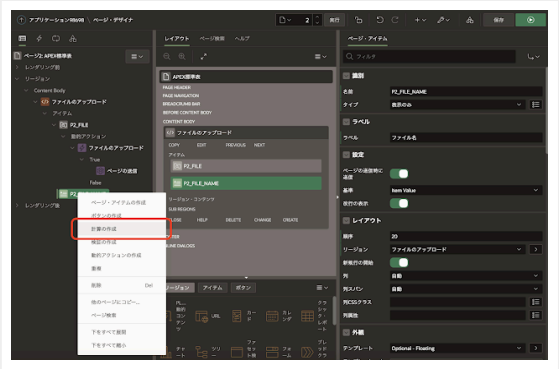
この時点でファイルのアップロードまでは実装できています。

アップロードされた結果を確認するために、アップロードされたファイルのファイル名を表示するページ・アイテムを追加します。

ページ・アイテムを作成し、**名前**を**P2_FILE_NAME**、**タイプ**は**表示のみ**、**ラベル**を**ファイル名**とします。

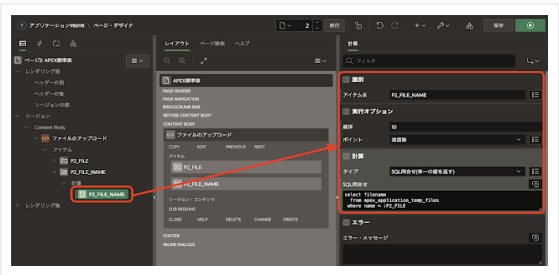


ファイルのアップロード後、表APEX_APPLICATION_TEMP_FILESから登録されたファイル名を取り出し、ページ・アイテムP2_FILE_NAMEに設定します。ページ・アイテムP2_FILE_NAMEに**計算の作成**を行います。



実行オプションの**ポイント**に**送信後**(送信後に計算を行うという意味)を選択します。**計算**の**タイプ**として**SQL問合せ(単一の値を返す)**を選択し、以下の**SQL問合せ**を設定します。

```
select filename
  from apex_application_temp_files
where name = :P2_FILE
```



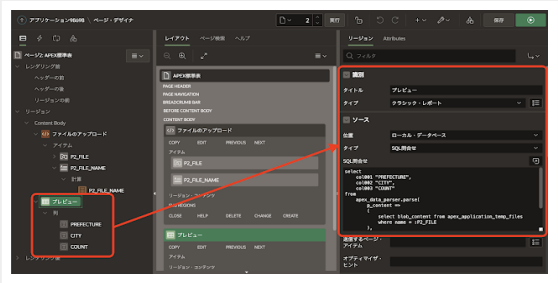
以上でアップロードされたファイルのファイル名が、ページ・アイテムP2_FILE_NAMEに表示されます。さらに、アップロードされたデータをプレビューします。

クラシック・レポートのリージョンを追加します。

リージョンを作成し、**名前**を**プレビュー**、**タイプ**を**クラシック・レポート**とします。**ソース**の**位置**は**ローカル・データベース**、**タイプ**を**SQL問合せ**とし、以下のSELECT文を記載します。

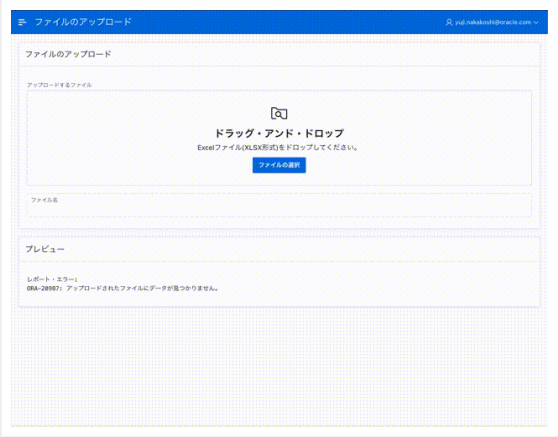
```
select
```

```
    col001 "PREFECTURE",
    col002 "CITY",
    col003 "COUNT"
from
    apex_data_parser.parse(
        p_content =>
        (
            select blob_content from apex_application_temp_files
            where name = :P2_FILE
        ),
        p_file_name => :P2_FILE_NAME,
        p_skip_rows => 1,
        p_file_charset => 'AL32UTF8',
        p_max_rows => 10
    )
```

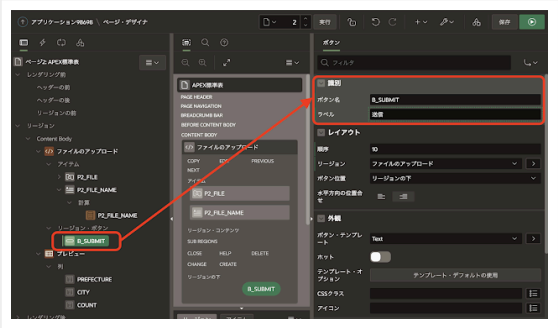


プレビューに使用したSELECT文ですが、この後に、このSELECT文の結果を表FUP_CITYLISTに挿入する処理を追加します。

ここまでの動作は以下のようになります。

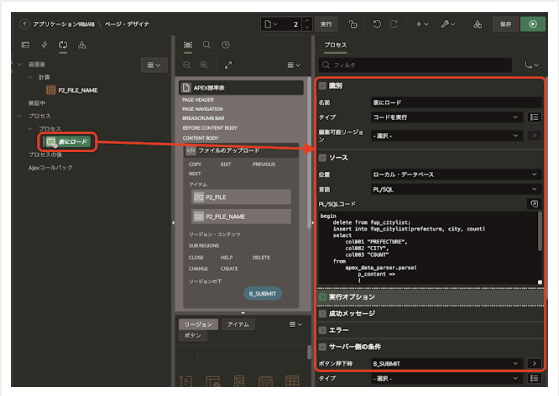


表FUD_CITYLISTにデータを投入するプロセスを作成します。データの送信を行うボタンを作成します。**ボタン名**を**B_SUBMIT**、**ラベル**を**送信**とします。



続いて、ボタンB_SUBMITが押された時に実行されるプロセスを作成します。**名前**を**表にロード**とし、**タイプ**は**コードを実行**、**ソース**の**位置**は**ローカル・データベース**、**言語**はPL/SQLを選択し、**PL/SQLコード**として以下を記述します(プレビューではないのでp_max_rowsの指定を除いています)。**サーバー側の条件**として、**ボタン押下時**にB_SUBMITを指定することにより、**送信**ボタンを押した時のみ実行されるようにします。

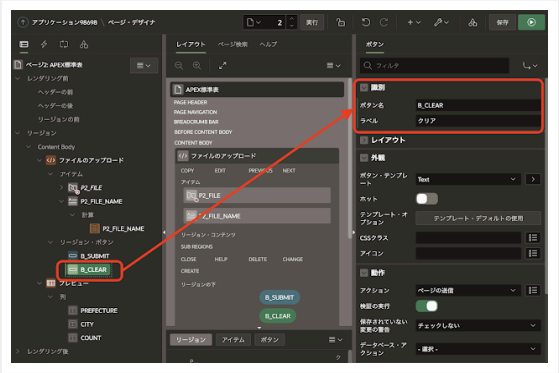
```
begin
    delete from fup_citylist;
    insert into fup_citylist(prefecture, city, count)
    select
        col001 "PREFECTURE",
        col002 "CITY",
        col003 "COUNT"
    from
        apex_data_parser.parse(
            p_content =>
            (
                select blob_content from apex_application_temp_files
                where name = :P2_FILE
            ),
            p_file_name => :P2_FILE_NAME,
            p_skip_rows => 1,
            p_file_charset => 'AL32UTF8'
        );
end;
```



上記のコードは、送信をクリックした際に、すでに保存されているデータをすべて削除し、新たにアップロードしたデータで入れ替えます。追加やマージといった動作にしたい場合は、PL/SQLのコードを変更することになります。

以上で表へのデータのロードも実行されるようになりました。動作確認をもう少し簡単にするため、ページを初期化するボタンと表FUD_CITYLISTの内容を表示するレポートを追加します。

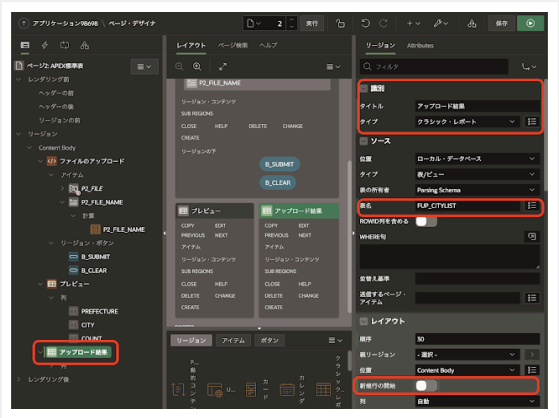
ボタンの作成を行い、**ボタン名**をB_CLEAR、**ラベル**を**クリア**とします。



ボタンB_CLEARを押した時に実行されるプロセスを作成します。作成したプロセスの**名前**を**ページの初期化**とし、**タイプ**に**セッション・ステートのクリア**を選択します。**サーバー側の条件**として、**ボタン押下時**に**B_CLEAR**を選択します。これでクリアのボタンを押した時に、ページが初期化されます。

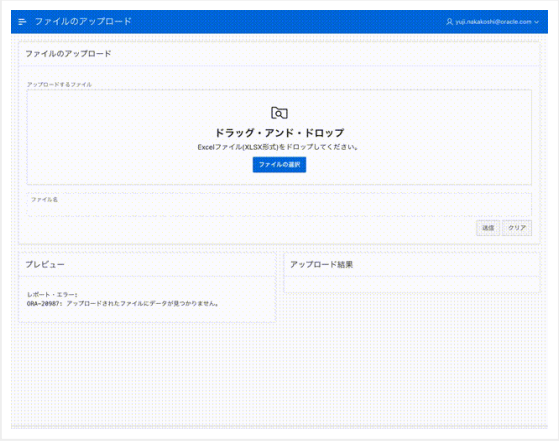
アップロード結果を表示するクラシック・レポートを作成します。

名前を**アップロード結果**とし、**タイプ**は**クラシック・レポート**を選択します。**ソース**の**表名**に**FUP_CITYLIST**を選択し、プレビューと横並びになるよう、**レイアウト**の**新規行の開始**を**OFF**とします。



以上で完成です。調整すべきところはありますが(例えばプレビューにORA-20987のエラーが発生するなど)、当初の目的は達成しています。

ページを実行し、実装を確認します。以下の動作になります。



作成済みの表のBLOB列を使う方法は、次の記事として記載します。

続く

Yuji N. 時刻: 18:23

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