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。

2020年7月22日水曜日

Oracle APEXでのデータ・ロード(6) - データのアンロード

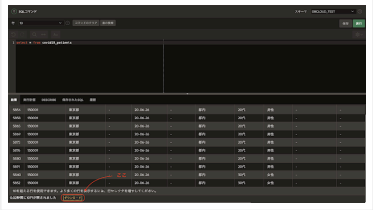
Oracle APEXでのデータ・ロードについて紹介してきましたが、最後に、ロードではなく、アンロード、つまり保存されているデータをCSVやJSONで出力する方法について紹介します。

データ・ワークショップ

データ・ワークショップのデータ・ロードの機能については、こちらの記事で紹介しています。データ・ワークショップには、データのロードの他に**データのアンロード**も行うことができます。

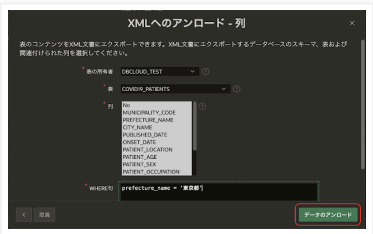
クリックすると、テキスト形式とXML形式の選択が求めらられます。

テキスト形式とありますが、実際はCSV形式です。CSV形式のダウンロードは、アプリケーションに組み込まれたレポートから実施することができるため、データ・ワークショップを積極的に利用する必要はあまりないでしょう。簡単にファイルに落としたいというのであれば、**SQLコマンド**の結果もCSV形式でダウンロードすることが可能です。



XML形式の出力はデータ・ワークショップでは出来ますが、レポートからはできません。ですので、この場合はデータ・ワークショップを使用します。

XML形式をクリックします。



表を最初に選択すると、指定可能な**列**が一覧されます。その中からアンロードの対象列を（複数）選択し、必要であれば**WHERE句**を指定します。ここでは**prefecture_name = '東京都'**を指定し、アンロードの対象を東京都に絞っています。すべて設定した後、**データのアンロード**をクリックすると、アンロード対象のデータがファイルとしてダウンロードされます。ファイル名は**指定した表名.xml**です。

ファイルの内容は以下のようなります。アンロードする列に指定されていてもデータが存在しなければ、XMLタグを含めて出力されません。

```
<?xml version="1.0"?>
<ROWSET>
 <ROW>
  <No>5854</No>
  <MUNICIPALITY_CODE>130001</MUNICIPALITY_CODE>
  <PREFECTURE_NAME>東京都</PREFECTURE_NAME>
  <PUBLISHED_DATE>2020-06-26T00:00:00.000</PUBLISHED_DATE>
  <PATIENT_LOCATION>都内</PATIENT_LOCATION>
  <PATIENT_AGE>20代</PATIENT_AGE>
  <PATIENT_SEX>男性</PATIENT_SEX>
  <PATIENT_LEFT_HOSPITAL>1</PATIENT_LEFT_HOSPITAL>
 </ROW>
 <ROW>
  <No>5858</No>
  <MUNICIPALITY_CODE>130001</MUNICIPALITY_CODE>
  <PREFECTURE_NAME>東京都</PREFECTURE_NAME>
  <PUBLISHED_DATE>2020-06-26T00:00:00.000</PUBLISHED_DATE>
  <PATIENT_LOCATION>都内</PATIENT_LOCATION>
  <PATIENT_AGE>20代</PATIENT_AGE>
  <PATIENT_SEX>男性</PATIENT_SEX>
  <PATIENT_LEFT_HOSPITAL>1</PATIENT_LEFT_HOSPITAL>
 </ROW>
 <ROW>
```

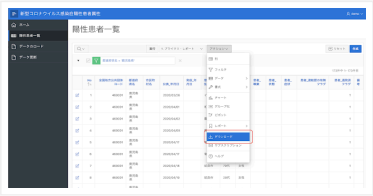
テキスト形式(CSV形式)では、**セパレータ**、**囲み文字(オプション)**、**列名を含める**のチェック、**ファイル形式**、**ファイルの文字セット**を指定できます。

残念なことに、なぜかapex.oracle.com上では、今回作成した表COVID19_PATIENTSをテキスト形式でアンロードしようとすると、エラーが発生します。原因は今のところ不明です。

レポートのダウンロード機能

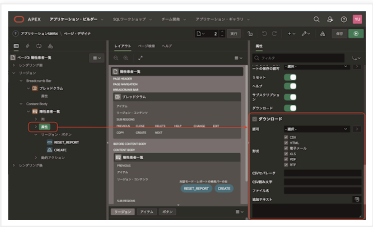
Oracle APEXが提供しているコンポーネントのうち、レポートを扱うものが３つあります。クラシック・レポート、対話モード・レポート、それと対話グリッドです。これらすべてのレポートに、データのダウンロード機能が提供されています。

対話モード・レポートを例にとって説明します。対話モード・レポートでは**アクション**・メニューに**ダウンロード**が含まれます。



ダウンロードを実行すると、**ダウンロード形式**の選択を要求されます。形式を選ぶとダウンロードが開始します。レポートで選択している列、および、行がダウンロードの対象になります。

ダウンロードの機能は、レポートの属性にて構成することができます。



出力可能な**形式**、**CSVセパレータ**、**CSV囲み文字**、**ファイル名**などはここで指定します。PDF 形式などは、あらかじめ外部のサーバーと連携するように構成されている必要があります。されていない場合は、形式として選んでダウンロードを実行するとエラーが発生します。そのため、ここのオプションを外しておきます。

データのロードの場合は、データ・ロード・ウィザードを構成するページを新規に作成し、アプリケーションに組み込む必要がありましたが、アンロード（ダウンロード）は、レポートに組み込みの機能であるため、レポートさえあれば追加の開発をする必要はありません。

RESTデータ・サービス

今まではOracle APEXが提供している機能を使い、まったくコードを書かずにデータのアンロードを行っています。最後にコードを記述することで、柔軟なデータ交換を実現する方法を紹介します。Oracle RESTデータ・サービスにて実装します。

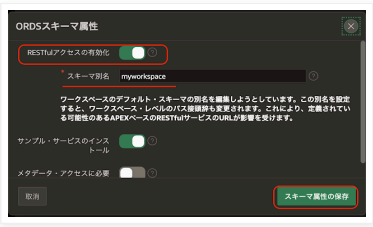
データ・ワークショップであれ、アプリケーションのレポート機能であれ、これらを使ってデータのアンロードを行うには、ユーザー認証が必要です。また、外部からのAPI呼び出しの結果、データを返すという実装はできません。RESTデータ・サービスでは、REST APIとしてデータのアンロードを実装します。

JSON形式でのアンロードを行うには、RESTデータ・サービスで実装する必要があります。

RESTデータ・サービスの実装には、**SQLワークショップ**の**RESTfulサービス**を使用します。

RESTデータ・サービスの利用が許可されていないと、**ORDSにスキーマを登録**というボタンが表示されます。

登録時のスキーマ属性の設定を要求されます。**スキーマ別名**はワークスペース名と同じ名前にします。なので、表示された値は変更しません。**サンプル・サービスのインストール**はON/OFFどちらでもかまいませんが、サンプルを参考にしたい場合はONにします。**メタデータ・アクセスに必要な認可**は、とりあえずOFFにします。これは後で変更できます。これらを設定して、**スキーマ属性**の保存をクリックします。



以上で、RESTデータ・サービスを実装する準備が整いました。

モジュールの作成

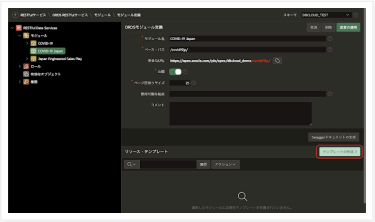
これから作成するREST APIをまとめる単位である、**モジュールの作成**を実行します。

モジュール名は任意です。ここでは**COVID-19 Japan**としています。日本語の入力は不可です。**ベース・パス**としては**/covid19jp/**を設定しています。ベース・パスはREST APIのURLに含まれることになります。**公開**は**ON**にしています。本来であれば、APIが出来上がってから公開した方がよいでしょう。これで**モジュールの作成**を実行します。

ページ区切りサイズは、この後に出てくるハンドラの設定のソース・タイプをCollection Queryを選択した際に、一回のリクエストで返却する行数になります。Collection Queryにはページネーションの仕組みが実装されています。

テンプレートの作成

モジュールが作成されたら、それに含まれるテンプレートを作成します。CSV形式のデータをダウンロードするためのテンプレートを作成します。**テンプレートの作成**をクリックします。

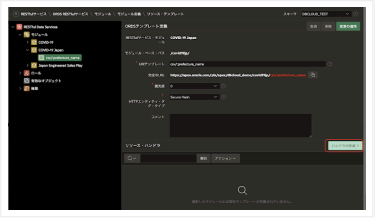


URIテンプレートとして、**csv/:prefecture_name**を設定し、**テンプレートの作成**を実行します。

最後尾が**/covid19jp/csv/都道府県名**となるURLがREST APIの呼び出し先になります。このURLにたいして、HTTPのGETメソッドを発行してCSVデータのダウンロードを行います。

ハンドラの作成

テンプレートを作成した後、**ハンドラの作成**を実行します。



メソッドは**GET**を選択します。CSVを出力するには、**ソース・タイプ**に**PL/SQL**を選択します。以前はQueryを選んで、書式としてCSVを選択することができましたが、最近のバージョンではDeprecatedとなっています。つまり、CSV形式での出力は標準機能からは外されています。JSONでの出力が標準機能となっています。

ソース・タイプとしてPL/SQLを選んで、出力形式も含めてコーディングします。ですので、書式がJSONとなっていて変更できませんが、こちらの設定を気にする必要はありません。ページ区切りサイズもPL/SQLの出力に影響を与えません。

CSV形式でデータを出力するコードとして、ソースに以下を入力します。

```
declare
  type cur_t is ref cursor;
  c cur_t;
  r_patient covid19_patients%rowtype;
begin
  -- 都道府県の指定を参照して、出力対象を絞り込む。無ければ全件出力する。
  if :prefecture_name is not null then
    open c for select * from covid19_patients
       where prefecture_name = :prefecture_name order by "No";
  else
    open c for select * from covid19_patients order by municipality_code, "No";
  end if;
  -- 出力がCSV形式であること、UTF-8であることをHTTPヘッダーで宣言する。
  owa_util.mime_header('text/csv', FALSE, 'utf-8');
  htp.p('Content-disposition: attachment; filename="patients' ||
       utl_url.escape(:prefecture_name, false, 'utf-8') || '.csv"');
  owa_util.http_header_close;
  -- BOM(バイトオーダーマーク)を最初に出力する。
  htp.prn(unistr('\feff'));
  -- カラムヘッダーを出力する。
  htp.prn('"No",');
  htp.prn('"全国地方公共団体コード",');
  htp.prn('"都道府県名",');
  htp.prn('"市区町村名",');
  htp.prn('"公表_年月日",');
  htp.prn('"発症_年月日",');
  htp.prn('"患者_居住地",');
  htp.prn('"患者_年代",');
  htp.prn('"患者_性別",');
  htp.prn('"患者_職業",');
  htp.prn('"患者_状態",');
  htp.prn('"患者_症状",');
  htp.prn('"患者_渡航歴の有無フラグ",');
  htp.prn('"患者_退院済フラグ",');
  htp.p('"備考"');
  -- データを一行づつ出力する。
  loop
    fetch c into r_patient;
    exit when c%notfound;
    htp.prn(r_patient."No"); htp.prn(',');
    htp.prn(to_char(r_patient.municipality_code,'000000')); htp.prn(',');
    htp.prn(r_patient.prefecture_name); htp.prn(',');
```

```
    htp.prn(r_patient.city_name); htp.prn(',');
    htp.prn(to_char(r_patient.published_date,'YYYY-MM-DD')); htp.prn(',');
    htp.prn(to_char(r_patient.onset_date,'YYYY-MM-DD')); htp.prn(',');
    htp.prn(r_patient.patient_location); htp.prn(',');
    htp.prn(r_patient.patient_age); htp.prn(',');
    htp.prn(r_patient.patient_sex); htp.prn(',');
    htp.prn(r_patient.patient_occupation); htp.prn(',');
    htp.prn(r_patient.patient_status); htp.prn(',');
    htp.prn(r_patient.patient_symptom); htp.prn(',');
    htp.prn(r_patient.patient_travel_history); htp.prn(',');
    htp.prn(r_patient.patient_left_hospital); htp.prn(',');
    htp.p(r_patient.remark);
  end loop;
  close c;
end;
```

作成したREST APIをGETで呼び出すと、以下のような結果が得られます。

```
"No","全国地方公共団体コード","都道府県名","市区町村名","公表_年月日","発症_年月日","患者_居住地","患者_年代","患者_性別","患者_職業","患者_状態","患者_症状","患者_渡航
1, 130001,東京都,,2020-01-24,,湖北省武漢市,40代,男性,,,,,1,
2, 130001,東京都,,2020-01-25,,湖北省武漢市,30代,女性,,,,,1,
3, 130001,東京都,,2020-01-30,,湖南省長沙市,30代,女性,,,,,1,
4, 130001,東京都,,2020-02-13,,都内,70代,男性,,,,,1,
5, 130001,東京都,,2020-02-14,,都内,50代,女性,,,,,1,
6, 130001,東京都,,2020-02-14,,都内,70代,男性,,,,,1,
7, 130001,東京都,,2020-02-15,,都内,80代,男性,,,,,1,
```

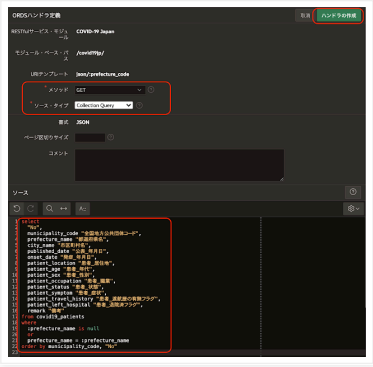
JSONの出力

JSONを出力する場合は、ソース・タイプのCollection Queryが使えるので、ソースの記述はより簡単です。

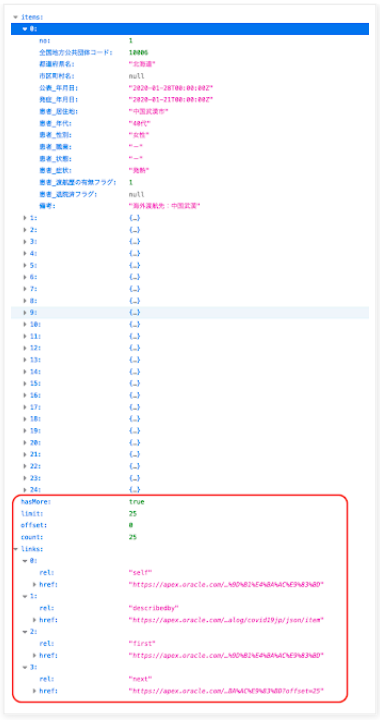
新たにテンプレートを作成し、**URIテンプレート**を**json/:prefecture_name**とします。作成したテンプレートにハンドラを作成します。

メソッドは**GET**を選択し、**ソース・タイプ**は**Collection Query**とします。**ソース**として以下のSQLを入力し、ハンドラの作成を行います。JSON形式へのフォーマットはOracle APEX側(正確にはOracle REST Data Services)が行うため、単にSELECT文をソースへ記述します。

```
select
  "No",
  municipality_code "全国地方公共団体コード",
  prefecture_name "都道府県名",
  city_name "市区町村名",
  published_date "公表_年月日",
  onset_date "発症_年月日",
  patient_location "患者_居住地",
  patient_age "患者_年代",
  patient_sex "患者_性別",
  patient_occupation "患者_職業",
  patient_status "患者_状態",
  patient_symptom "患者_症状",
  patient_travel_history "患者_渡航歴の有無フラグ",
  patient_left_hospital "患者_退院済フラグ",
  remark "備考"
from covid19_patients
where
  :prefecture_name is null
  or
  prefecture_name = :prefecture_name
order by municipality_code, "No"
```



Collection Queryで設定したREST APIを呼び出すと、以下のような結果がJSONとして返されます。



一回のリクエストで返される行数(itemsの配列に含まれるオブジェクトの数)は、ページ区切りサイズになります。出力の最後にページネーションを行うためのURLが追加出力されます。

"hasMore": true,
"limit": 25,
"offset": 0,
"count": 25,
"links": [
 {
"rel": "self",
"href": "https://apex.oracle.com/pls/apex/dbcloud_demo/covid19jp/json/%E6%9D%B1%E4%BA%AC%E9%83%BD"
},
 {
"rel": "describedby",
"href": "https://apex.oracle.com/pls/apex/dbcloud_demo/metadata-catalog/covid19jp/json/item"
},
 {
"rel": "first",
"href": "https://apex.oracle.com/pls/apex/dbcloud_demo/covid19jp/json/%E6%9D%B1%E4%BA%AC%E9%83%BD"
},
 {
"rel": "next",
"href": "https://apex.oracle.com/pls/apex/dbcloud_demo/covid19jp/json/%E6%9D%B1%E4%BA%AC%E9%83%BD?offset=25"
}
],

URLに引数としてoffsetが追加されています。すでに読み出した数をoffsetとして渡すことで、データ全体を複数回のAPI呼び出しに分けて取得することが可能になっています。

以上でOracle APEXが提供しているデータ・ロードの機能およびアンロードの機能紹介を終了します。

完

Yuji N. 時刻: 18:12

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