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。

2023年3月7日火曜日

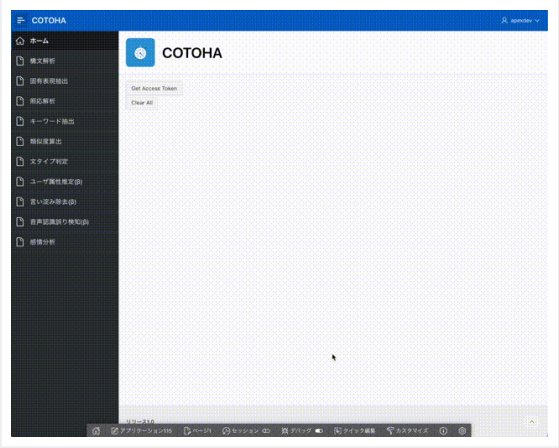
NTTコミュニケーションズのCOTOHA APIを呼び出してみる

NTTコミュニケーションズのCOTOHA API for Developersで使用できるAPIを、Oracle APEXのアプリケーションより呼び出してみます。

COTOHA API for Developers（無料）の範囲で使用できるAPIは、構文解析、固有表現抽出、照応解析、キーワード抽出、類似度算出、文タイプ判定、ユーザー属性推定(β)、言い淀み除去(β)、音声認識誤り検知(β)、感情分析です。

以下のようなアプリケーションを作成します。

APIの入力については、入力項目に対応するページ・アイテムを作成します。APIの応答はJSONを整形して表示します。それぞれのAPIの入力項目は似ているため、サービスごとのページの実装にそれほど違いはありません。



COTOHA API for Developersにアカウントを登録すると、**for Developersアカウント情報**として、**Developer Client id**と**Developer Client secret**が割り当てられます。APIの認証にこれらの値を使用します。

最初にWeb資格証明を作成します。

ワークスペース・ユーティリティより**Web資格証明**を開きます。

作成するWeb資格証明の**名前**は**COTOHA API Key**とします。**静的識別子**として**COTOHA_API_KEY**を指定します。

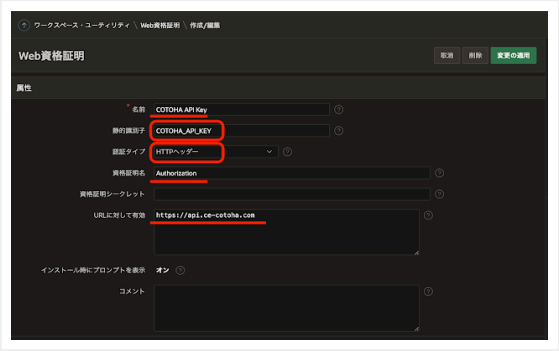
COTOHA APIのリファレンスのアクセストークン取得のページには、**認証方式(grantType)**として**client_credentials**を指定すると記載されているため、本来であればWeb資格証明の**認証タイプ**として**OAuth2クライアント資格証明フロー**が指定できるはずです。しかし、apex_web_service.make_rest_requestの引数**p_credential_static_id**にこのタイプの資格証明、引数**p_token_url**に**Access Token Publish URL**として与えられている以下のURLを指定しても、APIの認証に失敗します。

https://api.ce-cotoha.com/v1/oauth/accesstokens

APEXはclientIdとclientSecretはAuthorizationヘッダーのBasic認証の値として送信しますが、COTOHA APIはJSONドキュメントとして送信されることを要求しています。その違いが認証に失敗する原因と想定されます。

そのため、**認証タイプ**として**HTTPヘッダー**を選択し、**資格証明名**はHTTPヘッダー名である**Authorization**を指定します。**資格証明シークレット**は、**PL/SQLコード中でAccess Token Publish URLを呼び出して取得したアクセス・トークンを設定**します。作成画面では、資格証明シークレットの設定は不要です。

URLに対して有効は、**https://api.ce-cotoha.com**を指定します。

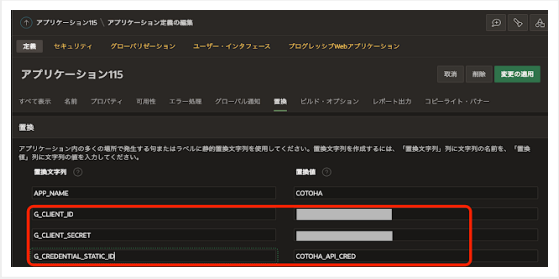


アプリケーション作成ウィザードを起動し、空のアプリケーションを作成します。作成したアプリケーションの**名前**は**COTOHA**としています。

アプリケーション定義の**置換**に、いくつか**置換文字列**を定義します。

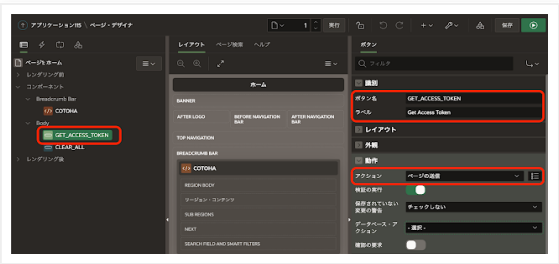
置換文字列**G_CLIENT_ID**の置換値として**Developer Client id**の値、**G_CLIENT_SECRET**として**Developer Client secret**の値を設定します。今回はAPIの検証が目的であり、API自体も無料で利用できる範囲であるため、これらの値を置換文字列として設定しています。一般の開発者は、これらの値を参照できない形で保存することが推奨です。

置換文字列**G_CREDENTIAL_STATIC_ID**の置換値として、**COTOHA_API_KEY**を設定します。



ホーム・ページにアクセス・トークンを取得するボタンを作成します。

識別の**ボタン名**は**GET_ACCESS_TOKEN**、**ラベル**は**Get Access Token**とします。**動作**の**アクション**はデフォルトの**ページの送信**です。



アクセス・トークンを取得するプロセスを作成します。アクセス・トークンの取得は、以下のPL/SQLコードで実施します。

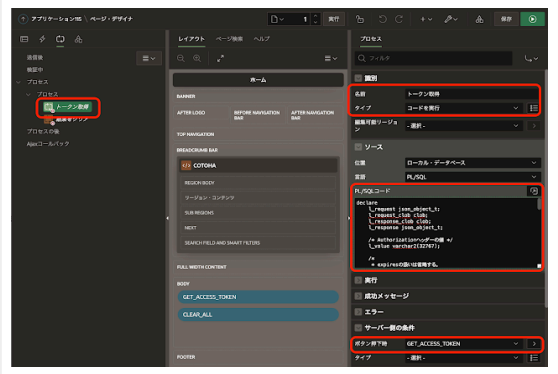
```
declare
    l_request json_object_t;
    l_request_clob clob;
    l_response_clob clob;
    l_response json_object_t;

    /* Authorizationヘッダーの値 */
    l_value varchar2(32767);

    /*
     * expiresの扱いは省略する。
    l_expires number;
    l_expire_date date;
    function unixtime_to_date(l_epochtime number)
    return date
    is
        l_date date;
    begin
        l_date :=  date'1970-01-01' + numtodsinterval( l_epochtime, 'SECOND');
        return l_date;
    end;
    */
begin
    l_request := json_object_t();
    l_request.put('grantType'    ,'client_credentials');
```

```
    l_request.put('clientId'    ,:G_CLIENT_ID);
    l_request.put('clientSecret',:G_CLIENT_SECRET);
    l_request_clob := l_request.to_clob();
    apex_web_service.clear_request_headers;
    apex_web_service.set_request_headers('Content-Type','application/json');
    l_response_clob := apex_web_service.make_rest_request(
        p_url => 'https://api.ce-cotoha.com/v1/oauth/accesstokens'
        ,p_http_method => 'POST'
        ,p_body => l_request_clob
    );
    if apex_web_service.g_status_code not in (200,201) then
        apex_debug.info(l_response_clob);
        raise_application_error(-20001, 'failed to get access token '
            || apex_web_service.g_status_code);
    end if;
    l_response := json_object_t(l_response_clob);
    l_value := 'Bearer ' || l_response.get_string('access_token');
    /*
     * expiresの扱いは省略。
    l_expires := to_number(trunc(l_response.get_string('issued_at')/1000))
        + to_number(l_response.get_string('expires_in'));
    l_expire_date := unixtime_to_date(l_expires);
    apex_debug.info('expires = %s', to_char(l_expire_date, 'YYYY-MM-DD HH24:MI:SS'));
    */
    /* Authorizationヘッダーの設定 */
    apex_credential.set_session_credentials(
        p_credential_static_id => :G_CREDENTIAL_STATIC_ID
        ,p_username => 'Authorization'
        ,p_password => l_value
    );
end;
```

get-access-token-cotoha.sql hosted with ❤️ by GitHub view raw

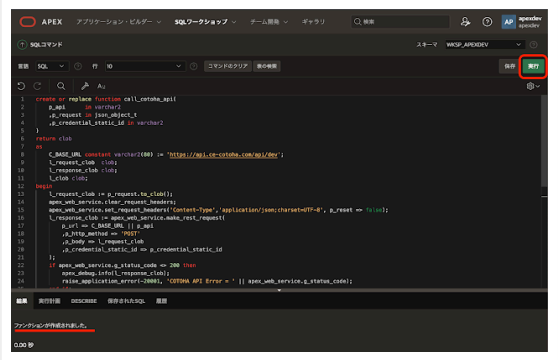


expires_inで指定されている秒数が経過すると取得したアクセス・トークンは無効になるため、更新する必要があります。今回のアプリケーションは、アクセス・トークンの更新までは実装していません。

COTOHA APIを呼び出すラッパーとなるファンクションを作成します。以下のコードをSQLコマンドより実行すると、ファンクションCALL_COTOHA_APIが作成されます。

```
create or replace function call_cotoha_api(
    p_api      in varchar2
    ,p_request in json_object_t
    ,p_credential_static_id in varchar2
)
return clob
as
    C_BASE_URL constant varchar2(80) := 'https://api.ce-cotoha.com/api/dev';
    l_request_clob  clob;
    l_response_clob clob;
    l_clob clob;
begin
    l_request_clob := p_request.to_clob();
    apex_web_service.clear_request_headers;
    apex_web_service.set_request_headers('Content-Type','application/json;charset=UTF-8', p_res
    l_response_clob := apex_web_service.make_rest_request(
        p_url => C_BASE_URL || p_api
        ,p_http_method => 'POST'
        ,p_body => l_request_clob
        ,p_credential_static_id => p_credential_static_id
    );
    if apex_web_service.g_status_code <> 200 then
        apex_debug.info(l_response_clob);
        raise_application_error(-20001, 'COTOHA API Error = ' || apex_web_service.g_status_code
    end if;
    select json_serialize(l_response_clob returning clob pretty) into l_clob from dual;
    return l_clob;
end;
```

call-cotoha-api.sql hosted with ❤️ by GitHub view raw



それぞれのAPIを呼び出す画面を作成します。

構文解析の画面は以下のような実装になります。

APIの入力となるページ・アイテムを作成します。**構文解析**では、**P1_SENTENCE**と**P1_TYPE**を作成しています。APIを呼び出すボタン**SUBMIT**、APIの出力を表示するページ・アイテム**Px_RESULT**は、すべてのページにあります。

APIを呼び出す**プロセス**の**タイプ**として**APIの呼び出し**を選択します。**設定**の**タイプ**に**PL/SQL Procedure or Function**を選択し、**プロシージャまたはファンクション**として先ほど作成した**CALL_COTOHA_API**を選びます。

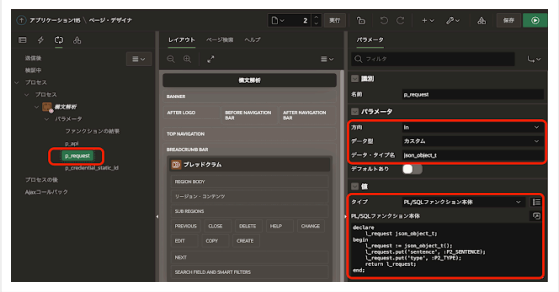
パラメータの**ファンクションの結果**の**値**として、**P2_RESULT**を選択します。ページ番号は変わりますが、どのCOTOHA APIでもページ・アイテム**Px_RESULT**が**ファンクションの結果**を保持します。

引数**p_api**の**値**の**タイプ**として**静的値**を選択し、**静的値**として、**ベースURLの部分を除いたAPIのURL**を指定します。

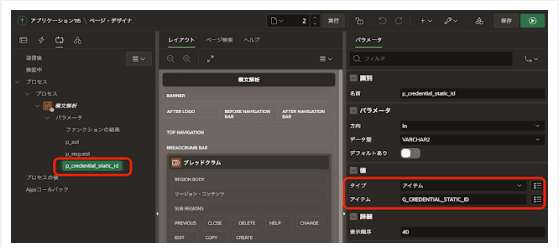
引数**p_request**の**値**の**タイプ**として**PL/SQLファンクション本体**を選択し、**PL/SQLファンクション本体**として以下のコードを記述します。ページ・アイテムの値を属性として含んだJSON文書を作成します。

```
declare
    l_request json_object_t;
begin
    l_request := json_object_t();
    l_request.put('sentence', :P2_SENTENCE);
    l_request.put('type', :P2_TYPE);
    return l_request;
end;
```

cotoha-json-input.sql hosted with ❤ by GitHub view raw



引数**p_credential_static_id**として、置換文字列の**G_CREDENTIAL_STATIC_ID**を指定します。



引数p_requestを作成するコードやp_apiのURLは、呼び出すCOTOHA APIのサービスごとに変更して、構文解析以外のサービスを呼び出すページを作成しています。

COTOHA APIを呼び出すOracle APEXのアプリケーションは以上で完成です。

今回作成した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/cotoha-api.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: 16:01

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