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3月18日水曜日

sentryファンクションによる外部ユーザー認証

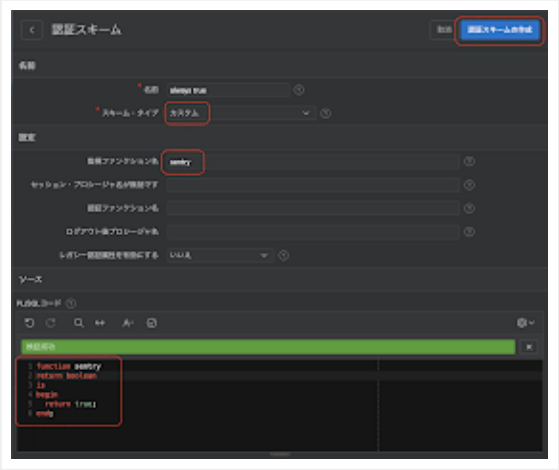
Oracle APEXの開発メンバーの一人、Christian Neumuellerによるこちらの記事に記載されている、外部の認証をOracle APEXに引き継ぐ方法を紹介します。この記事ではOracle Formsとの連携を意識していますが、例ではsqlplusを使用しています。Oracle FormsやOracle E-Business Suiteが稼働している環境が手元にある方はそれほど多くはいないはずなので、適切な選択でしょう。ユーザー認証の実装にはAPEX_JWTパッケージと、カスタム認証スキームのsentryファンクションを使用しています。

以前に**認証と認可の実装を学ぶ**というテーマでOracle APEXが持っている認証と認可を行う仕組みについて勉強会を実施しました。資料はこちらのスライドになります。このときは**監視ファンクション(sentryファンクション)**について、ほとんど説明をしませんでした。このファンクションはセッション管理およびユーザー認証を自力で実装する際に使用します。今まで参照した情報で、一番詳しくて、かつ、わかりやすい説明はMenno Hoogendijk（当時はオランダのQualogyというところに勤めていましたが、現在はオラクルに所属しているOracle APEXのプロダクト・マネージャーです)によるExploring the details of APEX sessionsだと思います。詳しく理解したい場合には、お勧めの資料です。

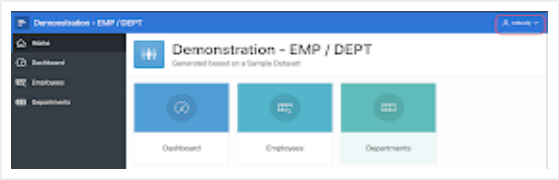
では、sentryファンクションから始めましょう。辞書からは、歩哨とか見張り、という意味になっていますが、Oracle APEXでsentryファンクションが行うことは、受け取ったHTTPリクエストが**有効なセッションに含まれるかどうか**を真偽値で返すことです。trueであればセッションが有効ということでページの処理に入ります。有効なセッションがなければ新規にセッションが作成されます。falseであればセッションは有効ではないので、ページの処理には入らず、ユーザー認証を要求するかページへのアクセスを拒否します。

sentryファンクションの動作を理解するために、つねにtrueを返すファンクションを定義してみます。**共有コンポーネント**の**認証スキーム**を開いて、新たに認証スキームを作成します。認証スキームの作成ウィザードが開いたら、**スキームの作成**として**ギャラリからの事前構成スキームに基づく**を選択します。認証スキームの設定が開いたら、**名前**を設定し(ここではalways true)、**スキームタイプ**は**カスタム**、**監視ファンクション名**としてsentry、そして以下のコードをソースの**PL/SQLコード**として指定します。設定した後、**認証スキームの作成**をクリックすると、認証スキームが新規に作成され、それがカレント・スキームに設定されます。

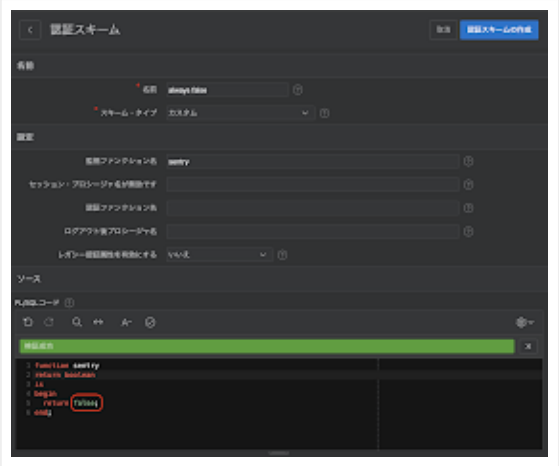
```
function sentry
return boolean
is
begin
  return true;
end;
```



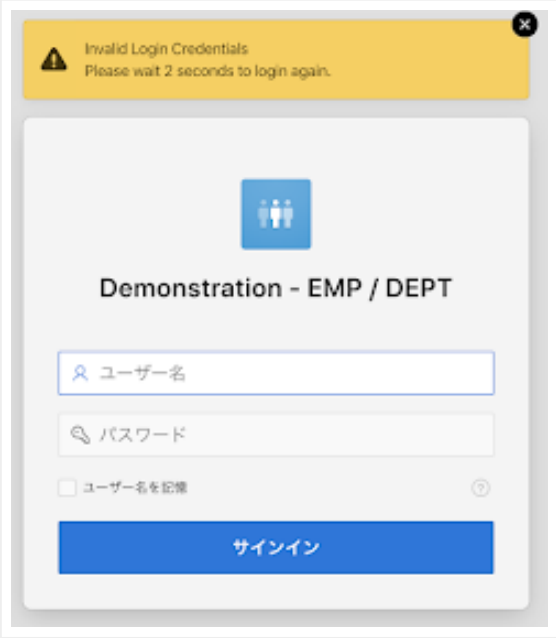
この認証スキームを実装したアプリケーションは、ユーザー認証なしでアクセスできます。また、有効なセッションが無い(URLに有効なセッションIDが含まれない)場合は、つねに新しいセッションが作成されます。ユーザー認証を行っていないため、ユーザー名はnobodyになっています。



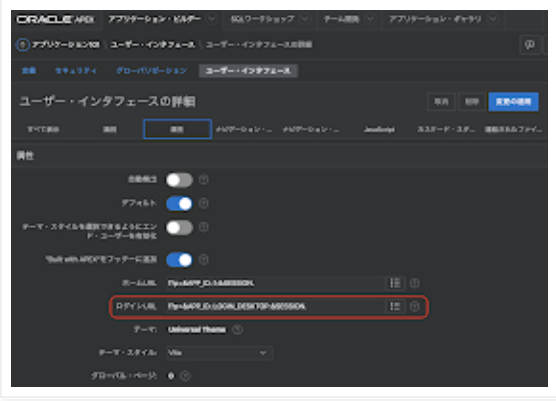
次に、sentryファンクションが必ずfalseを返す認証スキームを作成し、カレント・スキームにします。作り方は先ほどと同じです。



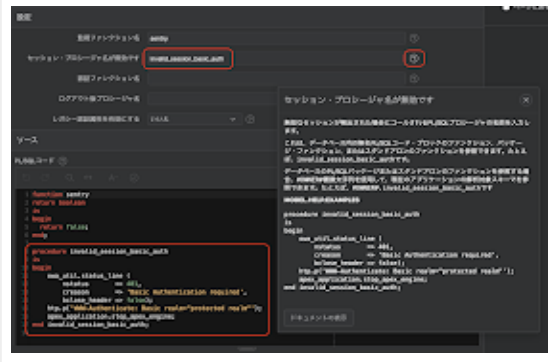
この状態でアプリケーションを実行すると、sentryファンクションが必ずfalseを返すため、標準のユーザー認証画面が表示されます。ただし、どのようなユーザー名やパスワードを入力しても、**そのユーザーが認証されることはありません**。sentryファンクションは、ユーザー認証されたかどうかを判断してtrueを返すコードを含まず、必ずfalseを返すためです。



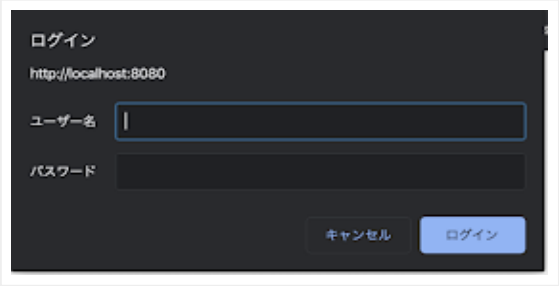
標準のログイン・フォームと違うフォームを使用する、または、標準をカスタマイズするには、**ユーザー・インターフェースの詳細**の**属性**に含まれる**ログインURL**として指定されているページを置き換えるか、指定されているページを変更します。



フォームという仕組み自体を置き換えるには、**セッション・プロシージャ名が無効です**を設定します。例えば、ヘルプをクリックして表示されるプロシージャーinvalid_session_basic_authを設定してみます。



ログインのフォームの代わりに、HTTPのベーシック認証を要求されるようになります。



この他にも、ユーザー認証にまつわるカスタマイズを実装する色々な仕掛けがありますが、それらを最初に制御するのはsentryファンクションです。ここでtrueと返せばアプリケーションにアクセスは許可されますし、falseを返せばアクセスは禁止されます。ですので、sentryファンクションに記載されるロジックにて、ユーザーがどのように認証されるかを決めることができます。Oracle APEXが標準で提供している機能をほぼ使用せず、最も自由度の高い実装を可能にしますが、生産性という意味では一番低くなります。なので、**本当に必要でなければ、sentryファンクションは使わない方がよいです**。

さて、最初に紹介した、こちらの記事ですが、JWT(JSON Web Token)を作成し、それを渡すことで認証情報の引き継ぎを行っています。JWTを受け取ったOracle APEXのアプリケーションでは、sentryファンクション内で受け取ったJWTの検証を実施しています。

IDトークン(JWT)が問い合わせ変数X01として渡されることを前提として記述されたsentryファンクションが以下になります。元のコードから認証と関係していない部分を削除し、コメントを日本語で付加しています。問い合わせ変数のX01はOracle APEXのAjaxコールの際に使われることが想定されており、これはチェックサムによる保護の対象になっていないため、IDトークンの受け渡しとして使用しています。電子署名(正確にはHMAC)の検証を行うために使用している鍵情報として"**sharedkey!**"を使っています。この鍵はJWTの発行者と対象の間のみで共有します。

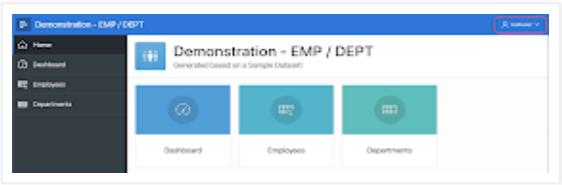
```
function sentry
   return boolean
is
   l_x01      varchar2(32767);
   l_jwt      apex_jwt.t_token;
   l_jwt_user varchar2(255);
begin
   /* 問い合わせパラメータX01としてJWTが渡される。*/
   l_x01 := v('APP_AJAX_X01');
   apex_debug.trace('X01=%s_x01');
   if l_x01 like '%.%.%' then -- header.payload.signatureのフォーマット確認
      begin
         /* JWTをデコードする。HMACの検証も行う */
         l_jwt := apex_jwt.decode (
                     p_value        => l_x01,
                     p_signature_key => sys.utl_raw.cast_to_raw('sharedkey!') );
         apex_debug.trace('JWTload=%s', l_jwt.payload);
         /* IDトークンの発行者(iss)、対象(aud)および有効期限の検証 */
         apex_jwt.validate (
            p_token => l_jwt,
            p_iss   => 'sqlplus',
            p_aud   => 'APEX' );
         apex_debug.trace('...validated');
         /* JWTをパースし、ユーザーの一意識別子(sub)を取り出す */
         apex_json.parse (
            p_source => l_jwt.payload );
         l_jwt_user := apex_json.get_varchar2('sub');
      exception when others then
         apex_debug.trace('...error: %s', sqlerrm);
```

```
      end;
    end if;
    /*
     * JWTがX01で渡されている場合は、認証が要求されている
     * セッションが継続していることを前提にした場合、X01にJWTは渡されないため、
     * すぐに以下の処理が呼び出される。
     */
    if apex_authentication.is_public_user then -- セッションがパブリック・ユーザーだから未ログイン
      if l_jwt_user is not null then -- JWTを受け取って、すでにユーザー名が取り出されている
        apex_authentication.post_login (
          p_username => l_jwt_user ); -- JWTに含まれるユーザーをAPP_USERとする
      else
        return false; -- JWTを受け取っていないので、拒否
      end if;
    elsif apex_application.g_user <> l_jwt_user then -- 認証済みユーザーとJWTのユーザーが異なる
      apex_debug.trace('...login user %s does not match JWT user %s',
        apex_application.g_user,
        l_jwt_user );
      return false; -- 違うユーザーなので拒否
    end if;
    return true; -- セッションを継続
end sentry;
```

このsentryファンクションで認証されるURLを生成するためのPL/SQLコードが以下になります。ホスト名やベースとなるURLは、それぞれの環境によって異なるでしょう。
http://localhost:8080/ords/xepdb1/f?p=101:1の部分です。URLに含まれるLEVEL9の部分は必須ではありませんが、デバッグ・レベルを9 - TRACEのレベルまで引き上げることで、sentryファンクションのデバッグを行えるようにしています。

```
set define off
set serveroutput on
declare
  l_jwt varchar2(32767);
begin
  l_jwt := apex_jwt.encode (
    p_iss          => 'sqlplus',
    p_aud          => 'APEX',
    p_sub          => 'TESTUSER',
    p_exp_sec      => 10,
    p_signature_key => sys.utl_raw.cast_to_raw('sharedkey!') );
 sys.dbms_output.put_line (
       'http://localhost:8080/ords/xepdb1/f?p=101:1:::LEVEL9:&x01='||l_jwt);
end;
/
exit;
```

印刷されたURLでアプリケーションにアクセスしたときに、ユーザー名がtestuserとして表示されれば、sentryファンクションが正しく機能しています。



p_exp_secとして10が設定されていますから、ここで生成されたトークンが有効なのは生成後１０秒までです。

完

Yuji N. 時刻: 17:28

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