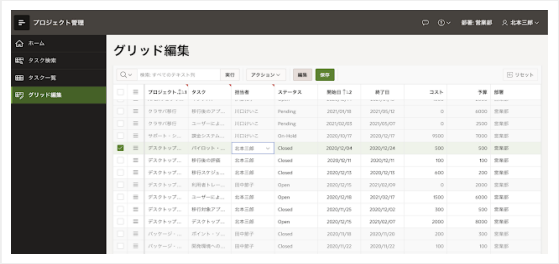
日日是Oracle APEX

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12月29日火曜日

アクセス制御の実装サンプル解説(1) - はじめに

アクセス制御の実装サンプルとして、こちらのアプリケーションを作ってみました。



よくデモに使用しているタスク一覧のCSVファイルを元に、アプリケーションを作成しています。

サンプルは複数のセッションの間で、干渉せずに更新できるようにしています(セッションIDにより分離している)。その部分は特殊なので省いて、データの操作で使われるファセット検索、対話モード・レポートとフォーム、対話グリッドで、どのようにアクセス制御を実装しているかを紹介しようと思います。

４本の記事を予定しています。

1. 初期アプリケーションの作成(この記事)
2. データ・ローディングの実装
3. ファセット検索ページの追加
4. 対話モード・レポートとフォームの追加
5. 対話グリッドの追加
6. 認可スキームについて

データ・ローディングを実装してCSVファイルを読み込んだ後に、データを操作する機能を提供するページを追加します。その際に、それぞれにアクセス制御を実装していきます。

作成するアプリケーションですが、以下の従業員でサインインします。

伊藤和子 - 営業部 - 管理者
川口けいこ - 営業部 - コントリビュータ
田中節子 - 営業部 -
北本三郎 - 営業部 -
木下愛 - 経理部 - 管理者
三浦康太 - 経理部 - コントリビュータ
山田太郎 - 経理部 -
高橋実 - 経理部 -
虎尾十兵衛 - 開発部 - 管理者
森花子 - 開発部 - コントリビュータ
大谷一朗 - 開発部 -
岡本一 - 開発部 -

- ロールが管理者であれば、すべてのタスクを参照することができる。
- ロールがコントリビュータであれば、自分が所属している部署に関するタスクの作成、更新、削除ができる。
- ロールのない従業員は、自分が所属している部署のタスクの参照と自分がアサインされているタスクの更新ができる（タスクの作成と削除は不可）。

以上の条件をそれぞれのページに実装します。

では、元になるアプリケーションを作成していきます。

クイックSQLによる表の作成

クイックSQLに以下の定義を与え、３つの表PAC_EMPLOYEES、PAC_PROJECTS、PAC_TASKSを作成します。

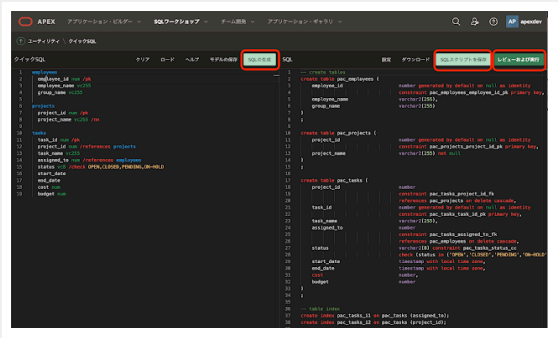
```
employees
  employee_id num /pk
  employee_name vc255 /nn
  group_name vc255 /nn

projects
  project_id num /pk
  project_name vc255 /nn

tasks
  task_id num /pk
  project_id num /references projects
  task_name vc255 /nn
  assigned_to num /references employees
  status vc8 /check OPEN,CLOSED,PENDING,ON-HOLD
  start_date
  end_date
  cost num
  budget num
```

DDLを生成する設定として、以下を指定します。

- オブジェクト接頭辞: PAC
- 主キー: ID列
- 日付データ型: TIMESTAMP WITH LOCAL TIME ZONE
- セマンティクス: デフォルト

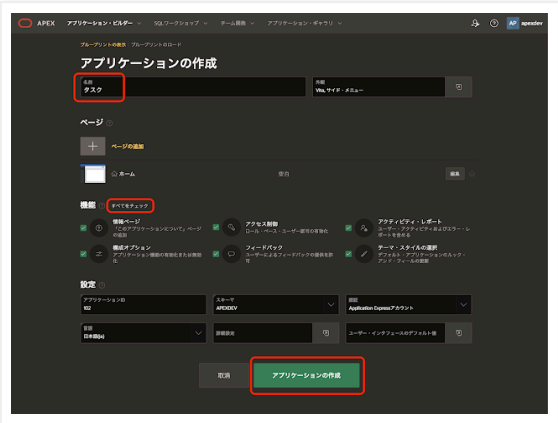
- 自動主キー: OFF
- 互換性: 18c以上
- 主キーに表名を接頭辞として付けます: OFF

SQLの生成、**SQLスクリプトの保存**、**レビューおよび実行**を順次実行し、生成されたDDLより表を作成します。最後にアプリケーションの作成を行うこともできますが、それは行いません。



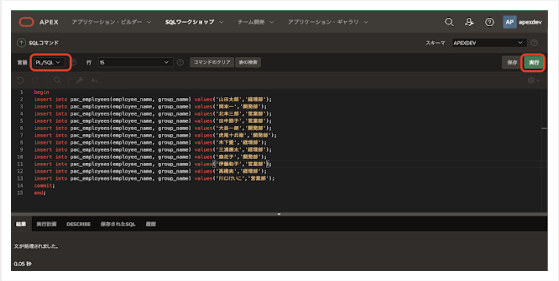
アプリケーションの作成

アプリケーション作成ウィザードを呼び出し、空のアプリケーションを作成します。**名前**は**任意**です。**機能**は**すべてをチェック**します（最低限、**アクセス制御**が必要です）。そして、**アプリケーションの作成**をクリックします。



アプリケーションが作成されたら、PAC_EMPLOYEES表に従業員データを投入します。**SQLワークショップ**の**SQLコマンド**から以下を実行します。

```
begin
insert into pac_employees(employee_name, group_name) values('山田太郎','経理部');
insert into pac_employees(employee_name, group_name) values('岡本一','開発部');
insert into pac_employees(employee_name, group_name) values('北本三郎','営業部');
insert into pac_employees(employee_name, group_name) values('田中節子','営業部');
insert into pac_employees(employee_name, group_name) values('大谷一朗','開発部');
insert into pac_employees(employee_name, group_name) values('虎尾十兵衛','開発部');
insert into pac_employees(employee_name, group_name) values('木下愛','経理部');
insert into pac_employees(employee_name, group_name) values('三浦康太','経理部');
insert into pac_employees(employee_name, group_name) values('森花子','開発部');
insert into pac_employees(employee_name, group_name) values('伊藤和子','営業部');
insert into pac_employees(employee_name, group_name) values('高橋実','経理部');
insert into pac_employees(employee_name, group_name) values('川口けいこ','営業部');
commit;
end;
```



投入されたデータは以下のSQLで確認できます。

```
select * from pac_employees
```

先ほど作成したアプリケーションのアプリケーションIDを確認します。

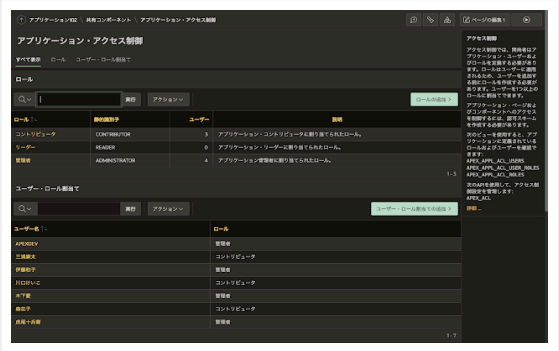
何人かのユーザーに、API呼び出しによってロールを割り当てます。

```
declare
  C_APP_ID constant number := <アプリケーションID>;
begin
  apex_acl.add_user_role(c_app_id, '虎尾十兵衛', 'ADMINISTRATOR');
  apex_acl.add_user_role(c_app_id, '木下愛', 'ADMINISTRATOR');
  apex_acl.add_user_role(c_app_id, '三浦康太', 'CONTRIBUTOR');
  apex_acl.add_user_role(c_app_id, '森花子', 'CONTRIBUTOR');
  apex_acl.add_user_role(c_app_id, '伊藤和子', 'ADMINISTRATOR');
  apex_acl.add_user_role(c_app_id, '川口けいこ', 'CONTRIBUTOR');
end;
```

ADMINISTRATORが管理者、CONTRIBUTORがコントリビュータのロールです。

この作業は作成したアプリケーションに組み込まれたアクセス制御の画面から行うこともできますし、共有コンポーネントのアプリケーション・アクセス制御からも操作することが可能です。

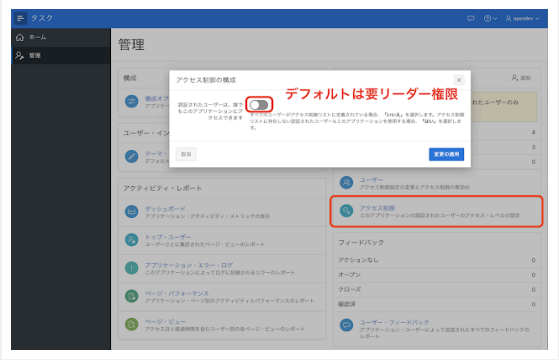
共有コンポーネントからの操作は、以下の画面から行います。



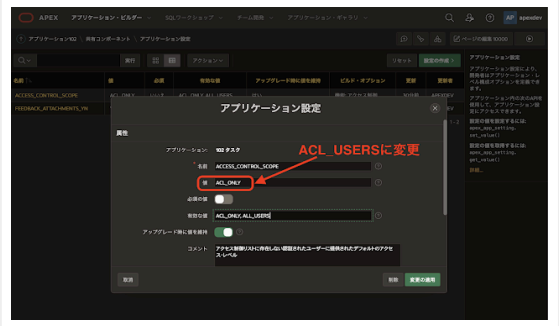
作成したアプリケーションに組み込まれているアクセス制御は以下です。

アプリケーション作成直後は、アプリケーションを作成した開発者アカウントだけが管理者ロールを持っています。アプリケーションにアクセス制御を組み込むと、アプリケーションにアクセスするために、ユーザーにはリーダー権限（ロール）が必須の状態がデフォルトになります。ですので、アプリケーション作成直後は、管理者以外のアクセスは拒否されます。

リーダー権限を不要にするには、アクセス制御の構成を変更する必要があります。アプリケーションからは以下の画面になります。今回はリーダー権限を不要にするため、以下に説明する操作を実施してください。



または、共有コンポーネントに含まれるアプリケーション設定の ACCESS_CONTROL_SCOPEを ACL_USERSへ変更します。



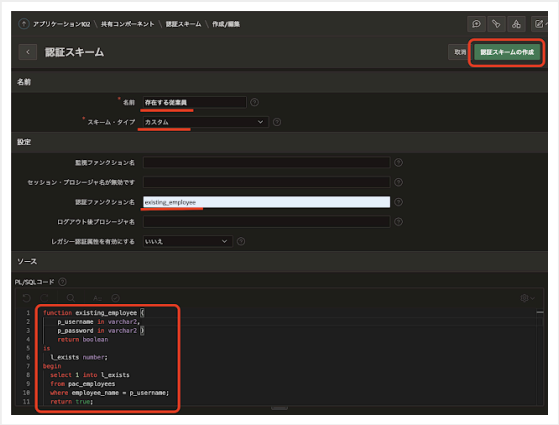
認証スキームの変更

登録された従業員がアプリケーションのユーザーとなるよう、**認証スキーム**を変更します。カスタム認証を作成します。認証に使用するPL/SQLコードは以下になります。従業員の存在だけを確認

し、パスワードは無視しています。あくまでもサンプルですので、そのまま使わないよう、お願いします。

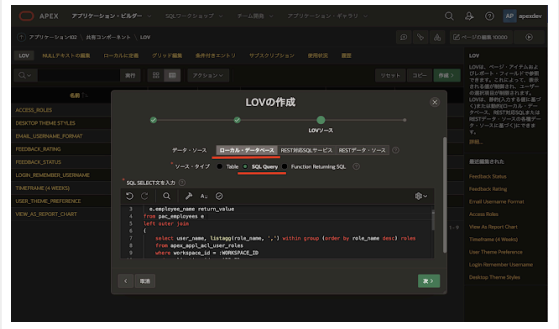
```
function existing_employee (
    p_username in varchar2,
    p_password in varchar2 )
    return boolean
is
  l_exists number;
begin
  select 1 into l_exists
  from pac_employees
  where employee_name = p_username;
  return true;
exception
  when no_data_found then
    return false;
end;
```

PL/SQLコードに登録した認証ファンクション**existing_employee**は、**認証ファンクション名**として設定し、**認証スキームの作成**を実行します。



これで認証スキームは作成したものに切り替わります。すでにサインイン済みのセッションは、一旦サインアウトする必要があります。

日本語の従業員名をユーザー名として指定するのは大変なので、ユーザー名を選択リストで選べるようにします。SQLによる動的なLOVを作成します。**共有コンポーネント**からLOVを開き、**タイプ**が**Dynamic**であるLOV、**名前**を**PAC_EMPLOYEES.EMPLOYEE_NAME_SIGNIN**として作成します。



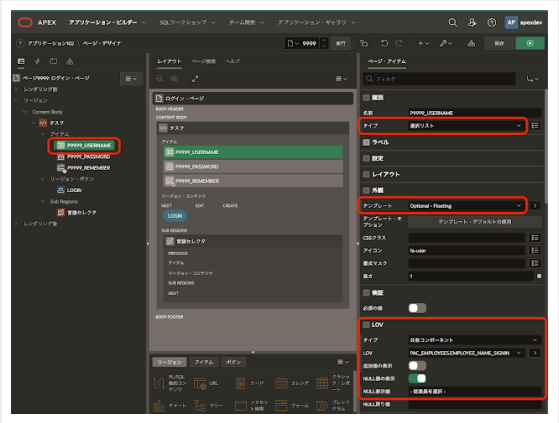
LOVに使用するコードとしては以下を指定します。

```
select
  e.employee_name || ' - ' || e.group_name || ' - ' || r.roles display_name,
  e.employee_name return_value
from pac_employees e
```

```
left outer join
(
   select user_name, listagg(role_name, ',') within group (order by role_name desc) roles
   from apex_appl_acl_user_roles
   where workspace_id = :WORKSPACE_ID
   and application_id = :APP_ID
   group by user_name
) r on e.employee_name = r.user_name
order by e.group_name, r.roles desc nulls last
```

従業員が所属している部署と割り当たっているロールを、一行で表示します。

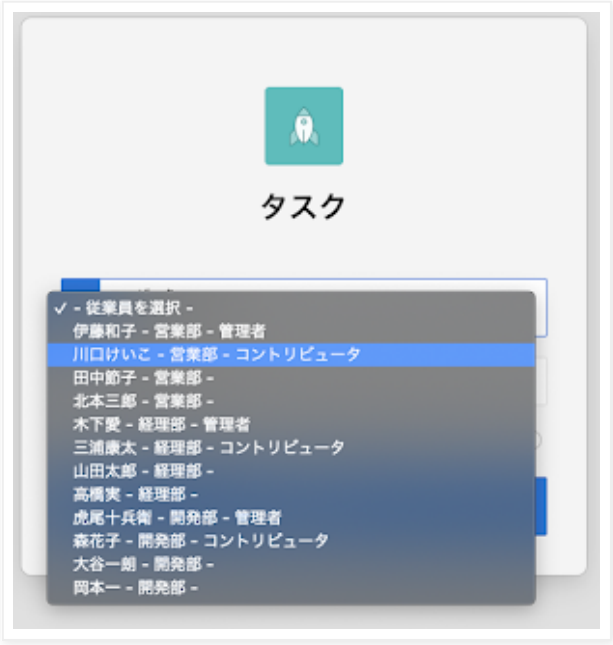
作成したLOVをログイン・ページに組み込みます。ログイン・ページをページ・デザイナで開き、**タイプ**を**選択リスト**、**テンプレート**を**Optional Floating**、LOVの**タイプ**として**共有コンポーネント**、LOVは先ほど作成した**PAC_EMPLOYEES.EMPLOYEE_NAME_SIGNIN**、**追加値の表示**は**OFF**、**NULL表示値**を**- 従業員を選択 -**とします。



保存を行い、アプリケーションを実行します。

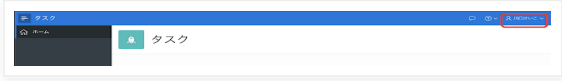
ログイン画面が変更されています。

選択リストから従業員を選びます。



従業員を選んでサインインします。

アプリケーションにアクセスできます。左上にサインインしたユーザー名が表示されます。



管理者ロールを持ったユーザーでサインインすると、管理メニューにアクセスできます。

この機能はアプリケーション作成時にチェックしていると、標準機能として組み込まれます。

アプリケーション・アイテムとアプリケーションの計算の登録

アプリケーションの雛形はほぼ出来上がっていますが、これからの開発を容易にするため、ユーザー認証直後にいくつかアプリケーション全体で使用可能な属性をアプリケーション・アイテムとして設定します。

以下の３つのアプリケーション・アイテムを登録します。

- G_GROUP_NAME = サインインしたユーザーが所属している部門名
- G_IS_ADMINISTRATOR = 管理者ロールを持っているとY、そうでない場合はN
- G_IS_CONTRIBUTOR = コントリビュータ・ロールを持っているとY、そうでない場合はN

共有コンポーネントの**アプリケーション・アイテム**を開き、**作成**をクリックします。**名前**として**G_GROUP_NAME**を指定し、**セッション・ステート保護**は必ず**制限付き - ブラウザからの設定不可**、とします。そして、**アプリケーション・アイテムの作成**を実行します。同様の作業を**G_IS_ADMINISTRATOR**、**G_IS_CONTRIBUTOR**についても実施します。

次に、これらのアイテムに値を設定するため、アプリケーションの計算を登録します。**共有コンポーネント**の**アプリケーションの計算**を開き、**作成**をクリックします。

最初にG_GROUP_NAMEの計算を登録します。**計算アイテム**として**G_GROUP_NAME**を選択します。**計算ポイント**は**認証後**です。**計算タイプ**は**SQL問合せ(単一の値を返す)**を選びます。計算として指定するSQLは以下になります。

```
select group_name from pac_employees where employee_name = :APP_USER
```

サインインしたユーザー名より、その所属部門を検索しています。

同様にして、G_IS_ADMINISTRATOR、G_IS_CONTRIBUTORも登録します。

G_IS_ADMINISTRATORは**計算タイプ**として、**ファンクション本体**を選びます。計算には以下のPL/SQLコードを指定します。

```
if apex_acl.has_user_role(
  p_role_static_id => 'ADMINISTRATOR'
) then
  return 'Y';
end if;
return 'N';
```

サインインしているユーザーが管理者ロールを持っているときに、Y、そうでないときにNが設定されるように、APEX_ACL.HAS_USER_ROLEファンクションを呼び出しています。

G_IS_CONTRIBUTORも同様ですが、p_role_static_idはCONTRIBUTORになります。

```
if apex_acl.has_user_role(
  p_role_static_id => 'CONTRIBUTOR'
) then
  return 'Y';
end if;
return 'N';
```

準備作業となるアプリケーションの作成は以上で完了です。

アプリケーションを実行し（すでにサインインしているセッションがあれば、一旦サインアウトし）、サインインを実施してみましょう。

設定内容を確認するため、**開発者ツール・バー**より**セッション**を実行します。**ビュー**に**アプリケーション・アイテム**を選択し、**設定**をクリックすると、今までに登録したアプリケーション・アイテムG_GROUP_NAME、G_IS_ADMINISTRATOR、G_IS_CONTRIBUTORに値が設定されていることが確認できるはずです。

次は、データ・ローディングを実装して、CSVファイルを表に取り込みます。

続く

Yuji N. 時刻: 21:43

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