TBS Engineering Dashboard Website

Installation Manual

Version 11/9/2019

Authors :

Mathieu Larivée, Oussama Lourhmati, Kha Pham, Motoki Shintani

Content

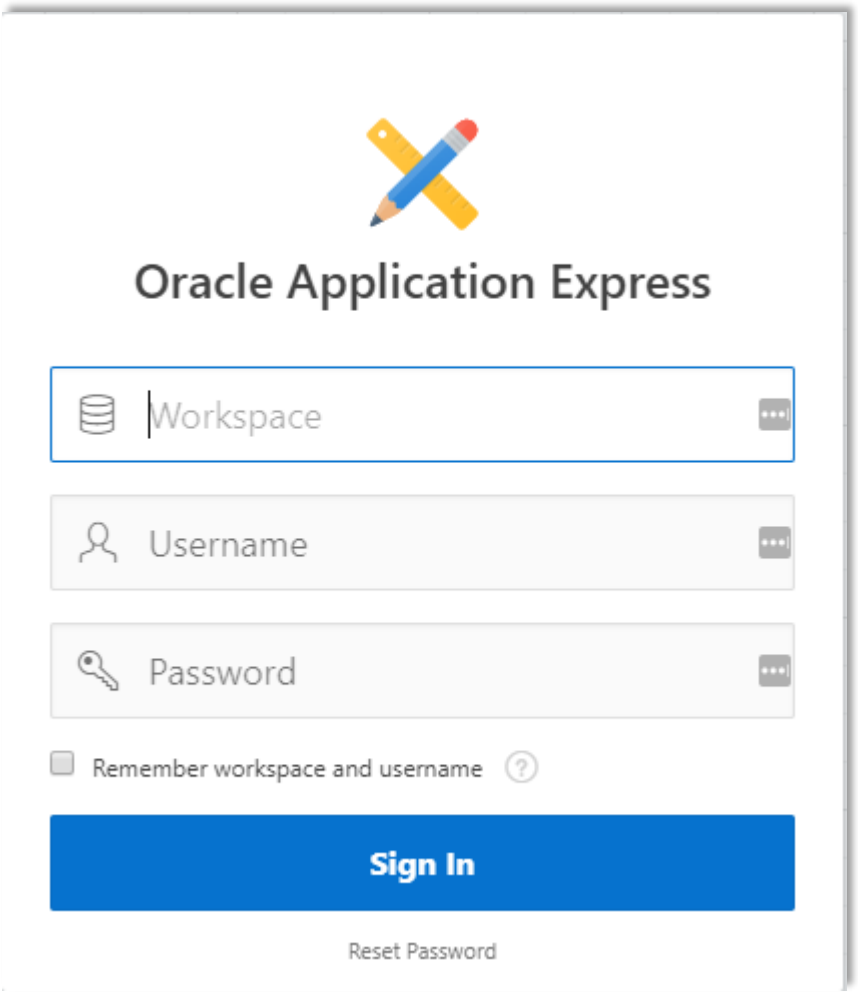
1. Introduction

This document serves as an installer for the TBS Engineering Dashboard Web Site. To install the tables, you have to accomplish the following tasks:

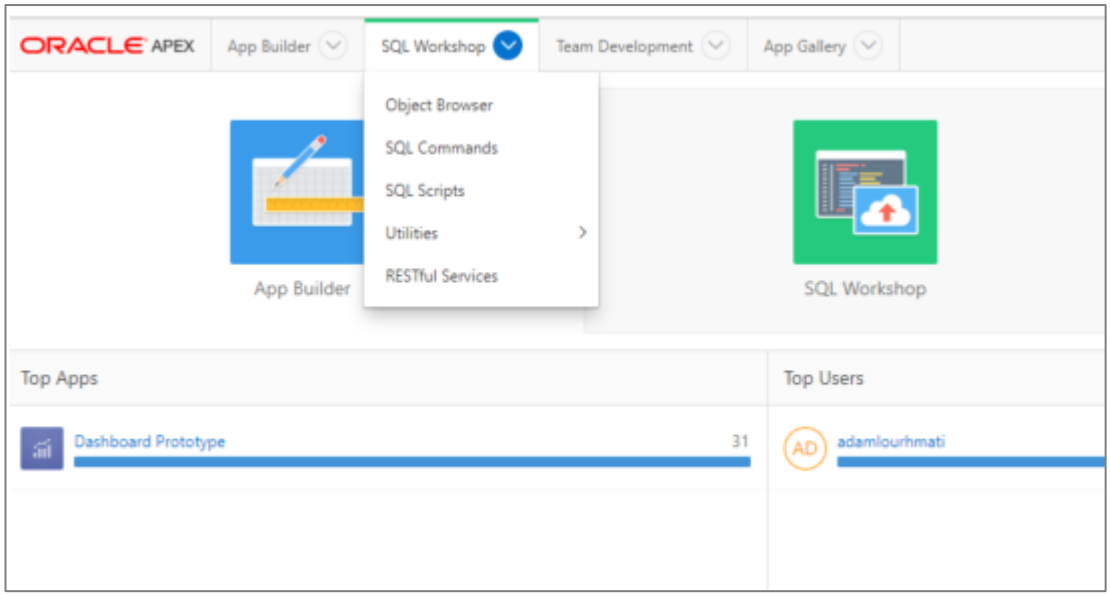
- Validate credentials to access the TBSCAD namespace;
- Drop, create and insert scripts of DB tables;
- Access to Oracle Application Express.

To configure the application, you have to follow the steps below:

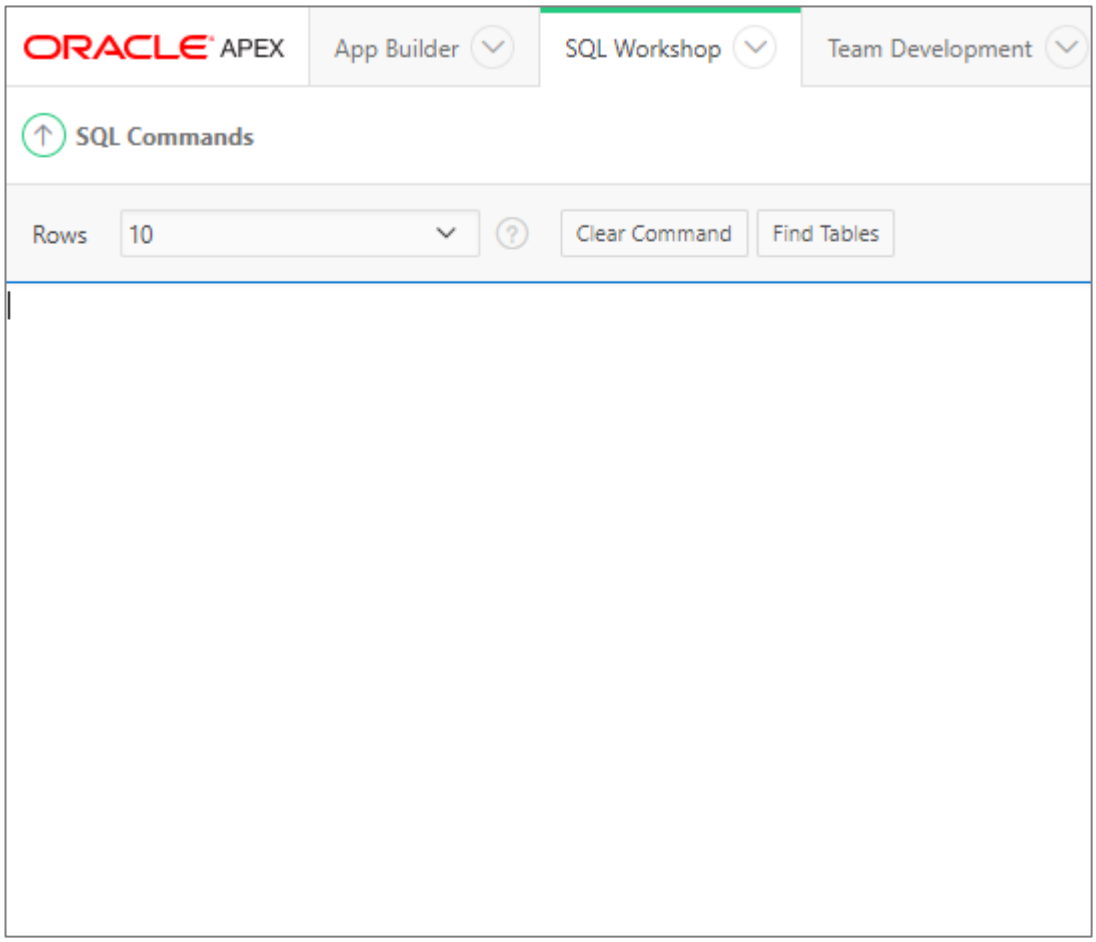
1. Access Oracle Application Express through the following link: http://apex.jbmlogic.com:8080/. And enter your credentials



2. After authentication step, you will then need to click on "**SQL Workshop**" menu, and select "**SQL Commands**".



3. This is the page where you will be able to paste all the following scripts.



The following sections only contain scripts that will be used to configure this application. Select the entire script with your mouse, copy it with CTRL + C, paste it with CTRL + V in the SQL commands of Oracle Application Express and press CTRL + ENTER to execute the script.

2. Drop Table/Package statements

If you are not reinstalling the tables and packages, you can skip to the next section, as this section's only purpose is to safely remove the tables and packages.

```
DROP TABLE APEX_SQL_QUERIES;

DROP TABLE USERS;

DROP TABLE ACL;

DROP TABLE PAGE;

DROP TABLE GSY_REJET_LOGIN;

DROP PACKAGE GSY_AUTH_APEX;

DROP PACKAGE SQL_QUERIES;
```

3. Create Table statements

3.1. Queries Table

The following script will create the table which will contain all SQL queries. Then, it will be possible to access to any SQL queries through a function. Each query has an **id** (query_id), a **content** which is the query itself, a **page number** where the query will be used and a **title** to identify the query.

```
CREATE TABLE APEX_SQL_QUERIES
(
    QUERY_ID NUMBER PRIMARY KEY,
    QUERY_CONTENT VARCHAR2(50000) NOT NULL,
    QUERY_PAGE_NUMBER NUMBER NOT NULL,
    QUERY_TITLE VARCHAR2(250) NOT NULL
)
```

3.2. Users table

The following script will create the table which will contain all the users. Each user has an **id** (**id_user**), **username**, a **password** which will be **hashed** and a **role** (A for admin, D for director).

```
CREATE TABLE USERS (
  ID_USER NUMBER GENERATED by default on null as
IDENTITY,
  USERNAME VARCHAR2 (15) NOT NULL UNIQUE,
  PASSWORD VARCHAR2 (4000) NOT NULL,
  ROLE VARCHAR2 (15),
  CONSTRAINT PK_USERS PRIMARY KEY (ID_USER)
);
```

3.3. Pages table

The table which will contain all pages will be created by executing the following script. For each page, there is an **ID** (**ID_PAGE**) which is also the page number, and a **description** to identify the page.

```
CREATE TABLE PAGE(
  ID_PAGE NUMBER,
  DESCRIPTION VARCHAR2 (4000),
  CONSTRAINT PK_PAGE PRIMARY KEY (ID_PAGE)
);
```

3.4. ACLs Table

To create the table of access control list (ACL), use the following script. For each entry in this table, there is an **id** (generated), the **authorisation number** (1 if the page is accessible, 0 if not), the **id** of the existing user and the **existing** page number.

```
CREATE TABLE ACL(
  ID_ACL NUMBER GENERATED by default on null as
IDENTITY,
  AUTHORISATION NUMBER,
  ID_USER NUMBER,
  PAGE NUMBER,
  CONSTRAINT PK_ACL PRIMARY KEY (ID_ACL),
  FOREIGN KEY (ID_USER) REFERENCES
USERS(ID_USER),
  FOREIGN KEY (PAGE) REFERENCES PAGE(ID_PAGE)
);
```

3.5. Logs Table

Table of logs will be created using the following script. Each failed access attempt will be logged in this table. For each entry, there is an **id** (generated), the username and the date of failed access attempt.

```
CREATE TABLE GSY_REJET_LOGIN
(
 ID_LOGIN NUMBER GENERATED by default on null as
IDENTITY,
 USERNAME VARCHAR2 (15) NOT NULL,
 DATE_LOG DATE NOT NULL,
 CONSTRAINT ID_LOGIN_PK PRIMARY KEY(ID_LOGIN)
);
```

4. Insert Into statements

4.1. Queries Insertion

The following link contains the insertion script of the SQL queries table

https://pastebin.com/5RyZwzRd

4.2. Access list fields insertion

The next link, contains the script that will be used to fill the access control list table

https://pastebin.com/GHzzd8ES

4.3. Pages fields insertion

This last link contains the insertion script to fill the pages table

https://pastebin.com/fwBXS4VH

5. Create Package statements

5.1. Package of SQL Queries

The following script will create a header of package which contains **get_sql_query_with_id** function used to access to SQL queries

```
create or replace PACKAGE sql_queries AS

  FUNCTION get_sql_query_with_id (p_query_id
apex_sql_queries.query_id%type) RETURN VARCHAR2;

END sql_queries;
```

The following link contains the body of this package

https://pastebin.com/58uxG6FX

5.2. Package of authentification & authorisation

The following script will create a header of package which contains

- **add_user** procedure used to add a user into the application, **get_hash** used to hash the user password
- **valid_user** used to validate user credentials
- **reject_login** used to loock user account after 3 failed login attemps
- **page_access** used to limit the user access to authorized pages according to his role and privileges

```
create or replace

PACKAGE GSY_auth_apex AS

  PROCEDURE add_user (p_username IN
VARCHAR2,p_password IN VARCHAR2, p_role IN
VARCHAR2);

  FUNCTION get_hash (p_username IN VARCHAR2,
p_password IN VARCHAR2)RETURN VARCHAR2;

  FUNCTION valid_user (p_username IN
VARCHAR2,p_password IN VARCHAR2)RETURN BOOLEAN;
```

```
    PROCEDURE reject_login (p_username IN
VARCHAR2);

    FUNCTION page_access (p_user IN
VARCHAR2,p_page IN NUMBER)  RETURN BOOLEAN;

END GSY_auth_apex;
```

Package body of **gsy_auth_apex** will be created by the script accessible by the following link

https://pastebin.com/HCsDUWQF

5.3. Add a user

Use the add_user function procedure to add users

```
BEGIN

 GSY_AUTH_APEX.add_user   (P_USERNAME   =>   'YOUR
USERNAME', P_PASSWORD => 'YOUR PASSWORD', P_ROLE
=> 'A') ;

END;
```