Contenu Afficher

Objet

Il est possible de manipuler des feuilles de calcul Google avec des APIs REST mais cela requiert un peu de codage pour la parsing du payload json.
— Ce POST n'est pas encore finalisé —

Normalement il est conseillé d'utiliser une API cliente selon le langage hôte (ph, python, nodejs…). Dans notre cas , PL/SQL, nous allons utiliser les appels REST natifs car il n'existe pas encore de client PL/SQL..

Pour lire une feuille Google Sheets, consulter la documentation *Basic Reading*: https://developers.google.com/sheets/api/samples/reading

Dans Oracle APEX, on pourrait utiliser une définition de REST Data Source mais le problème est que le *parser* ne reconnait pas les tableaux imbriqués dont voici un échantillon:
Exemple de *payload* renvoyé par l'API

```
{
"range": "'F2'!A2:D5",
"majorDimension": "ROWS",
"values": [
[
"test",
"4"
],
[
"test",
"3"
```

```
],
[
"test",
"2"
],
[
"zert",
"8"
]
]
}

1 error has occurred
Error during REST Data Source storage or discovery: ORA-40597: JSON
path expression syntax error ('$.$.values[*]') JZN-00217: Key step
contains unexpected characters at position 3
Found no data while parsing the service response.
```

	A	B	C	D	E	F
1	Horodateur	Adresse e-mail	Module	Descriptionduproblème	Votre Nom	Reponse
2	22/03/2022 18:07:41	patrick.monaco@gpmfactory.com	Option n° 1	reponse à q2 depuis F1		
3	22/03/2022 18:11:34	zzz@vision.eu		rep depuis f1	TEST	
4	22/03/2022 18:19:44	patrick.monaco@gpmfactory.com	Option n° 1	l'écran ne fonctionne plus		Réponse au problèr
5	22/03/2022 18:21:29	patrick.monaco@gpmfactory.com	Option n° 1	pb cycle		
6	22/03/2022 21:07:08	patrick.monaco@gpmfactory.com	Module Custom	Pb envoyé depuis F1	Patrick	Voic ma réponse u
7	26/03/2022 20:39:55	patrick.monaco@gpmfactory.com	Module 1	TEst	Pat	
8	22/03/2022 20:33:19	patrick.monaco@gpmfactory.com	M2	Pb ajouté par pmo	Patrick	pas de reponse enc
9	22/03/2022 20:52:29	patrick.monaco@gpmfactory.com	M1	test pb	Pat	
10	22/03/2022 20:56:42	patrick.monaco@gpmfactory.com	M2	pb sur M2	pat	
11	22/03/2022 21:05:19	patrick.monaco@gpmfactory.com	M3	Pb sur M3	Pat	
12	23/03/2022 19:04:00	patrick.monaco@gmail.com	M5	Pb sur chaînes	Pat	
13						
14						
15						
16						
17						
18						
19						
20						
21						

Réponses au formulaire 4 | F2 | Réponses au formulaire 1

Il faut donc se rabattre sur une approche manuelle. Celle qui est expliquée dans ce post consiste à effectuer un appel à la méthode GET dans un fonction PLSQL puis à retourner le contenu dans un pipe. Cela permet, au final, de manipuler la feuille à partir d'une requête SQL classique ou bien d'une vue.

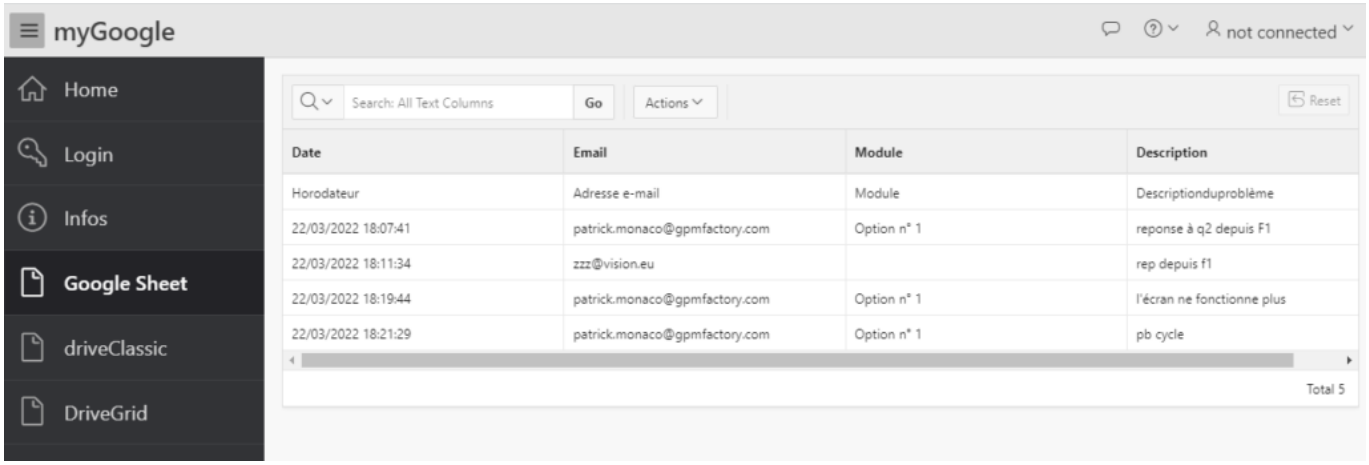
```
l_clob := apex_web_service.make_rest_request(
p_url => 'https://sheets.googleapis.com/v4/spreadsheets/' || tsheet ||
'/values/' || trange,
p_http_method => 'GET',
p_token_url => 'https://oauth2.googleapis.com/token',
p_credential_static_id => 'GoogleSAV'
);
```

Pour le range, il faut remplacer les espaces éventuels dans le nom de la feuille par un signe +

ex: trange varchar2(200) := 'Réponses+au+formulaire+4!A1:D5';

Création optionnelle d'une vue

```
CREATE OR REPLACE FORCE EDITIONABLE VIEW "GOOGLE_SHEET1_V" ("C1",
"C2", "C3", "C4", "C5", "C7", "C8", "C9", "C10") AS
select "C1","C2","C3","C4","C5","C7","C8","C9","C10"
from google_pkg.get_rows('Réponses+au+formulaire+4')
```



Contrôle d’accès.

Comme pour la plupart des accès à des données privées de Google, il faut utiliser le protocole OAUTH2.

Celui-ci est bien expliqué dans le document de Google.

Même si on ne peut pas utiliser les sourec REST, on peut néanmoins s’appuyer sur les *credentials* pour alléger la partie authentification. Sans cela, il faut récupérer un code d’accès et le convertir en *token*.

Pour Gogle Drive, l’intégration est plus simple car les REST data source sont opérationnels.
… mais, comme il ‘agit d’une donnée privée, il faut une authentification lors de la phase de design ‘redirection vers la page Google) qui n’est pas supportée par l’assistant. Donc, il faut auparavant générer manuellement un token valide qu’on fournira en paramètre de type Header Authorization Bearer <TOKEN>.
Le code pour générer ce token et decrit en annexe.

Une fois la découverte du *data profile* réalisée, on retire le paramètre *Authorization* qui devient superflu et on spécifie une autorisation basée sur un

des *credentials* enregistrés (*GoogleSAV* dans notre exemple). L'emploi du paramètre p_credential_static_id dans l'appel de apex_web_service.make_rest_request est très pratique car il soulage la tâche du développeur de toute la gestion du cycle de vie du token

ANNEXES

Obtention d'un token pour tester une REST data source

Le principe consiste à :

Constituer une URL contenant les parametres :

- response_code
- access_type
- redirect_url() . Cette url correspond à un module ORDS custom contenant un handler GET sur le *template* nommée *token*. Pour info, si on veut fourni un id de session qui sera retournée par Google, on ajoute le paramètre *state* qui contiendra ce qu'on veut bien y mettre.
- client_id
- Demande d'un code d'accès (dans une branch before Header d'une page afin de provoquer une redirection

Affichage d'un page d'authentification Google

Affichage optionnelle d'une page de consentement (envoyée par Google)

Redirection vers l'URL envoyée en paramètre

```
https://accounts.google.com/o/oauth2/auth?
scope=https://www.googleapis.com/auth/drive
&response_type=code
&access_type=offline
&redirect_uri=
&P0_REDIRECT.
&client_id=xxxxxxxxxxxxxx.apps.googleusercontent.com
```

Comme la string ne peut pas depasser 255 caractères, on préenregistre l'url de redirection dans une variable globale P0_REDIRECT que l'on mentionne avec la syntaxe &P0_REDIRECT.

Conversion du code d'accès en *token* (cf code source du package GOOGLE_PKG en annexe)

Affichage du token pour utilisation ultérieure. La durée de vie de vie est d'une heure environ. Ce token devra être fourni dans la variable de header nommée AUTHORIZATION et devra être précédé par le mot clef *Bearer* suivi d'un espace.

```
declare
l_token varchar2(500);
l_code varchar2(500);
begin
l_code := utl_url.unescape(GOOGLE_PKG.at_get_env_parameter('code')
);
l_token := GOOGLE_PKG.get_token(l_code);
htp.p('token=' || l_token);
htp.p('scope=');
```

```
htp.p(utl_url.unescape(GOOGLE_PKG.at_get_env_parameter('scope'))
);
end;
```

Author

- 

  Patrick

  GPM Factory