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11月8日水曜日

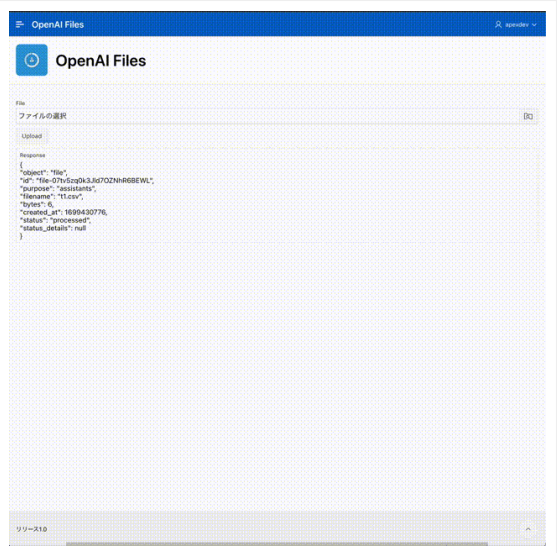
OpenAIのFiles APIを使ってファイルをアップロードする

OpenAIのFiles APIを呼び出して、ファイルの操作を行なうアプリケーションを作ってみました。

最近ベータ版が出た**Assistants API**に含まれる機能に**retrieval**と**code_interpreter**があります。これらの機能が使用する**assistant file**は、**purpose="assistants"**で**Files**にアップロードされているファイルの**file_id**を**assistant file**にアタッチするという仕様のようです。

Create assistant file
https://platform.openai.com/docs/api-reference/assistants/createAssistantFile

ファイルのアップロードができれば十分なので、あまり手の込んだ実装はしていません。



Downloadをクリックするとエラーが発生しました。エラー・メッセージを確認すると、そもそもpurposeがassistantsのファイルはダウンロード不可のようです。purposeがfine-tuneであればダウンロードできるのかもしれません。

```
body: {
  "error": {
    "message": "Not allowed to download files of purpose: assistants",
    "type": "invalid_request_error",
    "param": null,
    "code": null
  }
}
```

OpenAIのFiles APIを呼び出すパッケージ**UTL_OPENAI_FILES_API**を作成しています。上記のAPEXアプリケーションは、このパッケージに実装されたプロシージャを呼び出しています。クラシック・レポートのデータ・ソースだけはパッケージで実装せず、**RESTデータ・ソース**を定義しています。

```
create or replace package utl_openai_files_api
as

/**
 * OpenAI Upload fileを呼び出す。
 * 参照: https://platform.openai.com/docs/api-reference/files/create
 *
 * @param p_filename     アップロードされるファイル名
 * @param p_content_type ファイル形式
 * @param p_file_content ファイル本体
 * @param p_purpose      OpenAIでのファイルの用途 fine-tuneかassistantsのどちらか
 * @param p_credential_static_id Web資格証明
 * @param p_file アップロードされたファイルのメタデータ JSON形式
 * @return file_id
 */
function upload_file(
    p_filename      in varchar2
    ,p_content_type in varchar2
    ,p_file_content in blob
    ,p_purpose      in varchar2 default 'assistants'
    ,p_credential_static_id in varchar2 default 'OPENAI_API_KEY'
    ,p_file         out clob
)
return varchar2;

/**
 * OpenAI Retrieve file、Retrieve file contentを呼び出す。
 * 参照: https://platform.openai.com/docs/api-reference/files/retrieve
 * 参照: https://platform.openai.com/docs/api-reference/files/retrieve-contents
 *
 * @param p_file_id file_id
 * @param p_credential_static_id Web資格証明
 * @param p_file ファイルのメタデータ JSON形式
 * @param p_file_content ファイル本体 BLOB形式
 * @return file_id
 */
function retrieve_file(
    p_file_id       in  varchar2
    ,p_credential_static_id in varchar2 default 'OPENAI_API_KEY'
    ,p_file         out clob
    ,p_file_content out blob
)
return varchar2;

/**
 * OpenAI delete fileを呼び出す。
```

```
 * 参照: https://platform.openai.com/docs/api-reference/files/delete
 *
 * @param p_file_id file_id
 * @param p_credential_static_id Web資格証明
 */
function delete_file(
    p_file_id       in  varchar2
    ,p_credential_static_id in varchar2 default 'OPENAI_API_KEY'
    ,p_file out clob
)
return varchar2;

end utl_openai_files_api;
/

create or replace package body utl_openai_files_api
as

/**
 * ファイルのアップロード
 */
function upload_file(
    p_filename      in varchar2
    ,p_content_type in varchar2
    ,p_file_content in blob
    ,p_purpose      in varchar2
    ,p_credential_static_id in varchar2
    ,p_file         out clob
)
return varchar2
as
    l_response clob;
    l_multipart apex_web_service.t_multipart_parts;
    l_multipart_request blob;
    e_api_call_failed exception;
    l_file json_object_t;
begin
    apex_web_service.clear_request_headers;
    /*
     * アップロードするファイルを指定する。
     */
    apex_web_service.append_to_multipart(
        p_multipart     => l_multipart
        ,p_name         => 'file'
        ,p_filename     => p_filename
        ,p_content_type => p_content_type
        ,p_body_blob    => p_file_content
```

```
    );
    /*
     * purposeを指定する。
     */
    apex_web_service.append_to_multipart(
        p_multipart => l_multipart
        ,p_name => 'purpose'
        ,p_content_type => 'text/plain'
        ,p_body => p_purpose
    );
    l_multipart_request := apex_web_service.generate_request_body(l_multipart);
    l_response := apex_web_service.make_rest_request(
        p_url => 'https://api.openai.com/v1/files'
        ,p_http_method => 'POST'
        ,p_body_blob => l_multipart_request
        ,p_credential_static_id => p_credential_static_id
    );
    if apex_web_service.g_status_code <> 200 then
        apex_debug.info('status_code = %s, response = %s', apex_web_service.g_status_code, l_re
        raise e_api_call_failed;
    end if;
    p_file := l_response;
    l_file := json_object_t(p_file);
    return l_file.get_string('id');
end upload_file;

/**
 * ファイルのダウンロード
 */
function retrieve_file(
    p_file_id       in varchar2
    ,p_credential_static_id in varchar2
    ,p_file         out clob
    ,p_file_content out blob
)
return varchar2
as
    l_url varchar2(400);
    l_response clob;
    e_api_call_failed exception;
    l_file json_object_t;
begin
    /* ファイルのメタデータの取得 */
    l_url := 'https://api.openai.com/v1/files/' || p_file_id;
    apex_web_service.clear_request_headers;
    l_response := apex_web_service.make_rest_request(
        p_url => l_url
```

```
        ,p_http_method => 'GET'
        ,p_credential_static_id => p_credential_static_id
    );
    if apex_web_service.g_status_code <> 200 then
        apex_debug.info('status_code = %s, response = %s', apex_web_service.g_status_code, l_re
        raise e_api_call_failed;
    end if;
    /* ファイルのメタデータをp_fileとして返す。 */
    p_file := l_response;
    /* ファイルの内容をBOBとして取得する */
    l_url := l_url || '/content';
    apex_web_service.clear_request_headers;
    p_file_content := apex_web_service.make_rest_request_b(
        p_url => l_url
        ,p_http_method => 'GET'
        ,p_credential_static_id => p_credential_static_id
    );
    if apex_web_service.g_status_code <> 200 then
        apex_debug.info('status_code = %s, failed to download file', apex_web_service.g_status_
        raise e_api_call_failed;
    end if;
    l_file := json_object_t(p_file);
    return l_file.get_string('id');
end retrieve_file;

/**
 * ファイルの削除
 */
function delete_file(
    p_file_id       in varchar2
    ,p_credential_static_id in varchar2
    ,p_file         out clob
)
return varchar2
as
    l_url varchar2(400);
    l_response clob;
    e_api_call_failed exception;
    l_file json_object_t;
begin
    /* ファイルのメタデータの取得 */
    l_url := 'https://api.openai.com/v1/files/' || p_file_id;
    apex_web_service.clear_request_headers;
    l_response := apex_web_service.make_rest_request(
        p_url => l_url
        ,p_http_method => 'DELETE'
        ,p_credential_static_id => p_credential_static_id
```

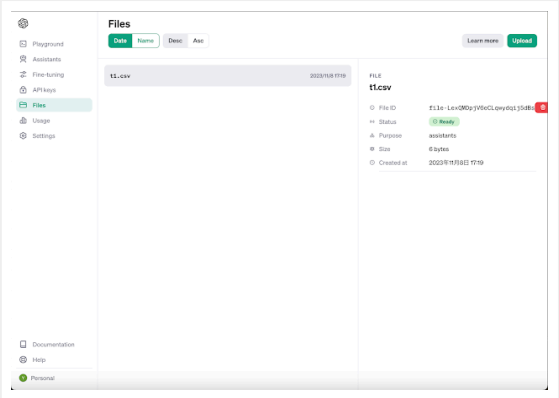
```
    );
    if apex_web_service.g_status_code <> 200 then
        apex_debug.info('status_code = %s, response = %s', apex_web_service.g_status_code, l_re
        raise e_api_call_failed;
    end if;
    /* 削除したファイルのメタデータをp_fileとして返す。 */
    p_file := l_response;
    l_file := json_object_t(p_file);
    return l_file.get_string('id');
end delete_file;

end utl_openai_files_api;
/
```

utl_openai_files_api.sql hosted with ❤️ by GitHub view raw

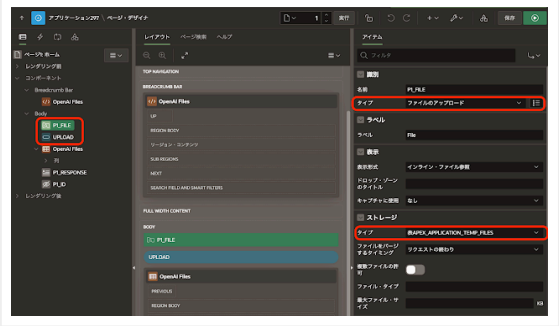
今回作成したAPEXアプリケーションのエクスポートを以下に置きました。
https://github.com/ujnak/apexapps/blob/master/exports/openai-files-sample.zip

アップロードしたファイルは、OpenAIの画面からも確認できます。



簡単に実装について紹介します。

ファイルのアップロードは、ファイルを選択するページ・アイテム**P1_FILE**（タイプは**ファイルのアップロード**）とボタン**UPLOAD**で実装しています。**ストレージ**の**タイプ**として**表APEX_APPLICATION_TEMP_FILES**を選択します。

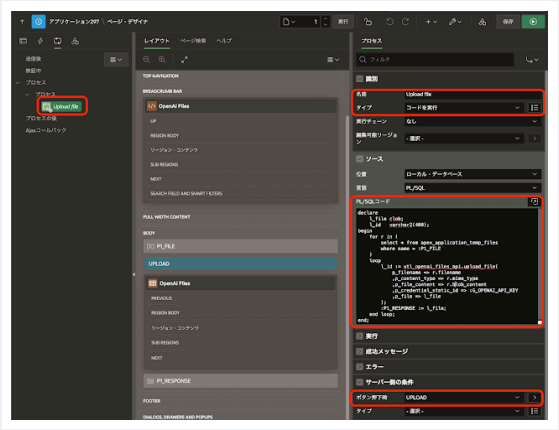


ボタン**UPLOAD**をクリックしたときに実行されるプロセスには、以下のコードを記述しています。

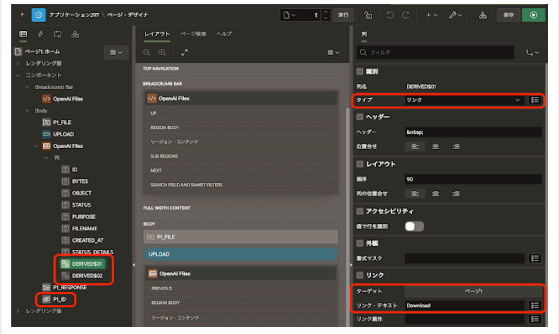
```
declare
```

```
    l_file clob;
    l_id   varchar2(400);
begin
    for r in (
        select * from apex_application_temp_files
        where name = :P1_FILE
    )
    loop
        l_id := utl_openai_files_api.upload_file(
            p_filename => r.filename
            ,p_content_type => r.mime_type
            ,p_file_content => r.blob_content
            ,p_credential_static_id => :G_OPENAI_API_KEY
            ,p_file => l_file
        );
        :P1_RESPONSE := l_file;
    end loop;
end;
```

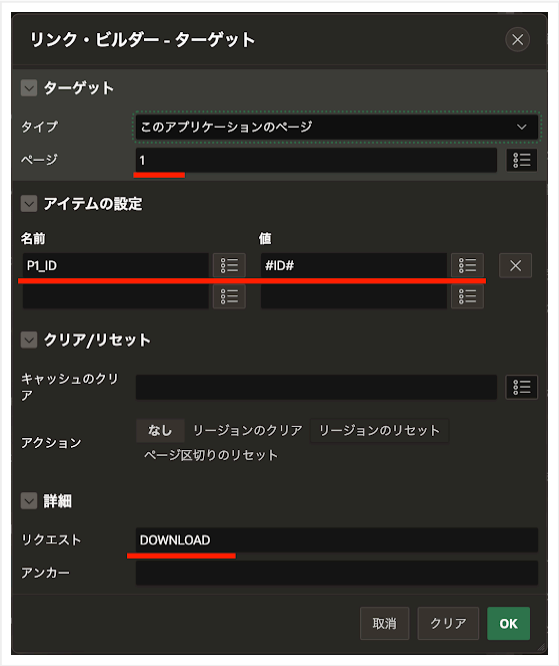
openai-upload-file.sql hosted with ❤️ by GitHub view raw



ファイルの**ダウンロード**、**削除**はクラシック・レポートに追加した**仮想列**に**リンク**を設定して**プロセス**を呼び出しています。



ターゲットの設定では**ページ**はページ**1**、つまり同じページに遷移させています。**file_id**をページ・アイテムの**P1_ID**に渡しています。**リクエスト**の値は、ダウンロードの場合は**DOWNLOAD**、削除の場合は**DELETE**を渡し、**レンダリング前**に作成している**プロセス**を呼び出す条件に使用します。

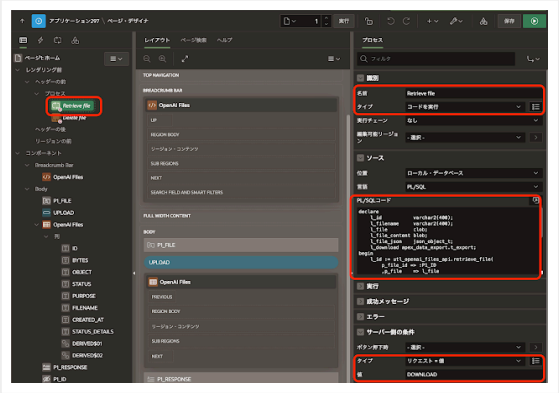


ファイルをダウンロードするプロセスは、**レンダリング前**の**ヘッダー前**に作成しています。実行するコードとして、以下を記述します。

```
declare
    l_id           varchar2(400);
    l_filename     varchar2(400);
    l_file         clob;
    l_file_content blob;
    l_file_json    json_object_t;
    l_download apex_data_export.t_export;
begin
    l_id := utl_openai_files_api.retrieve_file(
        p_file_id => :P1_ID
        ,p_file    => l_file
        ,p_file_content => l_file_content
        ,p_credential_static_id => :G_OPENAI_API_KEY
    );
    l_file_json := json_object_t(l_file);
    l_filename := l_file_json.get_string('filename');
    /* download */
    l_download.file_name := l_filename;
    l_download.as_clob := FALSE;
    l_download.content_blob := l_file_content;
    apex_data_export.download( p_export => l_download );
    apex_application.stop_apex_engine;
end;
```

openai-retrieve-file.sql hosted with ❤️ by GitHub　　view raw

サーバー側の条件の**タイプ**に**リクエスト = 値**を選択し、**値**として**DOWNLOAD**を設定します。

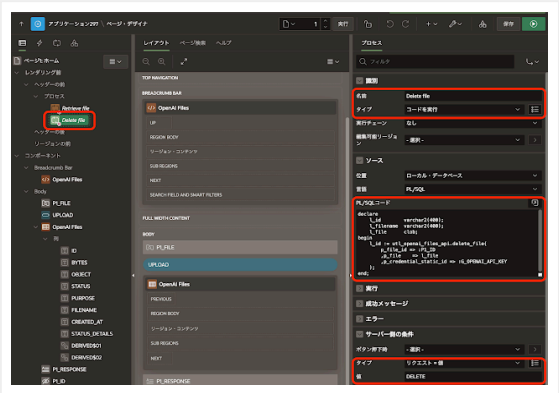


削除もほぼ同じ実装です。削除を行なうプロセスのコードとして、以下を記述します。

```
declare
    l_id        varchar2(400);
    l_filename  varchar2(400);
    l_file      clob;
begin
    l_id := utl_openai_files_api.delete_file(
        p_file_id => :P1_ID
        ,p_file     => l_file
        ,p_credential_static_id => :G_OPENAI_API_KEY
    );
end;
```

openai-delete-files.sql hosted with ❤️ by GitHub　　view raw

サーバー側の条件の**タイプ**に**リクエスト = 値**を選択し、**値**として**DELETE**を設定します。



OpenAIのFilesにアップロードされているファイルの一覧は、以下のURLを呼び出すことで取得できます。

https://api.openai.com/v1/files

最低１つでもファイルがアップロード済みであれば、RESTデータ・ソースの作成時にレスポンスを自動的に認識してくれます。

Id、Bytes、Object、Status、Purpose、Filename、Created_At、Status_Detailsが認識されています。

OpenAIのFiles APIを使ったAPEXアプリケーションの説明は以上です。

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: 18:07

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