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7月5日水曜日

TinyMCEでのイメージ・アップロード

Oracle APEX 23.1よりリッチ・テキスト・エディタの実装がCKEditorからTinyMCEに置き換えられました。今のところCKEditorも使用できますが、TinyMCEへ移行した方が良いことは間違いありません。

CKEditorをカスタマイズせずに使用している範囲であれば、**設定**の**ライブラリ**を**CKEditor**から**TinyMCE**に問題なく切り替えられるはずです。プラグインを使用してカスタマイズしている場合は、TinyMCEに同等のカスタマイズを行う必要があります。

CKEditorのイメージ・アップロードについては、以下の記事を書いています。

CKEditor5による画像アップロードを実装する(1) - 準備
CKEditor5による画像アップロードを実装する(2) - REST APIの作成
CKEditor5による画像アップロードを実装する(3) - 完成
CKEditor5による画像アップロードを実装する(4) - 画像をオブジェクト・ストレージに保存する
CKEditor5による画像アップロードを実装する(5) - 動的コンテンツを使う

TinyMCEを使用するにあたって、上記の実装の変更点について記述します。

CKEditorのイメージ・アップロードと同様に、Louis Moreauxさんのブログ記事 - Oracle APEX 23.1 - TinyMCE and image upload - を参考にしています。

APEXアプリケーションの変更

ページ番号２のフォームのページを変更します。

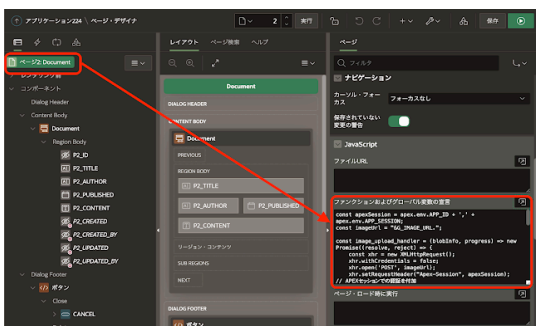
ページ・プロパティの**JavaScript**の**ファンクションおよびグローバル変数の宣言**で、images_upload_handlerを定義します。

images_upload_handlerについては、TinyMCE DocumentationのImage Uploads - images_upload_handlerで説明されています。

```
const apexSession = apex.env.APP_ID + ',' + apex.env.APP_SESSION;
const imageUrl = "&G_IMAGE_URL.";

const image_upload_handler = (blobInfo, progress) => new Promise((resolve, reject) => {
    const xhr = new XMLHttpRequest();
    xhr.withCredentials = false;
    xhr.open('POST', imageUrl);
```

```
    xhr.setRequestHeader("Apex-Session", apexSession);    // APEXセッションでの認証を付加

    xhr.upload.onprogress = (e) => {
        progress(e.loaded / e.total * 100);
    };

    xhr.onload = () => {
        if (xhr.status === 403) {
            reject({ message: 'HTTP Error: ' + xhr.status, remove: true });
            return;
        }

        if (xhr.status < 200 || xhr.status >= 300) {
            reject('HTTP Error: ' + xhr.status);
            return;
        }

        const json = JSON.parse(xhr.responseText);

        // json.locationからjson.urlへ変更。
        if (!json || typeof json.url != 'string') {
            reject('Invalid JSON: ' + xhr.responseText);
            return;
        }

        resolve(json.url + '?_apex_session=' + apexSession);  // APEXセッションでの認証を付加
    };

    xhr.onerror = () => {
        reject('Image upload failed due to a XHR Transport error. Code: ' + xhr.status);
    };

    const formData = new FormData();
    formData.append('file', blobInfo.blob(), blobInfo.filename());

    xhr.send(formData);
});
```

tinymce-images-upload-handler.js hosted with ❤️ by GitHub view raw



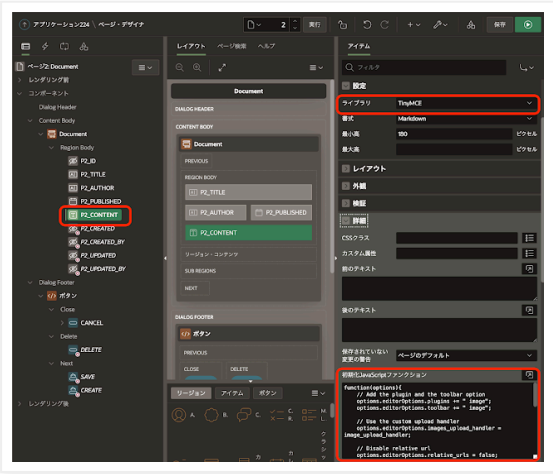
マニュアルに記載されているimages_upload_handlerの例に、APEXセッションで認証するためのコードを数行追加しています。また、TinyMCEのimages_upload_handlerの例では、画像アップロードのレスポンスが{ location: "画像のURL" }という形式で返されることが想定されていますが、今回の実装では{ url: "画像のURL" }なので、その点も修正しています。

TinyMCEのimageプラグインを有効にし、images_upload_handlerを設定します。

ページ・アイテム**P2_CONTENT**の**設定**の**ライブラリ**を**TinyMCE**に変更し、**詳細**の**初期化JavaScript ファンクション**に以下を記述します。

```
function(options){
    // Add the plugin and the toolbar option
    options.editorOptions.plugins += " image";
    options.editorOptions.toolbar += " image";

    // Use the custom upload handler
    options.editorOptions.images_upload_handler = image_upload_handler;

    // Disable relative url
    options.editorOptions.relative_urls = false;

    // Replace the editor content
    options.editorOptions.init_instance_callback = (editor) => {
        editor.setContent(editor.getContent());
    }

    // Define the URL converter and enable it
    options.editorOptions.convert_urls = true;
    options.editorOptions.urlconverter_callback = function(url, node, on_save, name) {
        console.log(url, node, on_save, name);
        if (node === "img" && name === "src" ){
            url = url.split("?")[0] + '?_apex_session=' + apexSession;
            console.log(url);
        }
        return url;
    }

    return options;
}
```

tinymce-initialize.js hosted with ❤️ by GitHub view raw



ここで、**urlconverter_callback**として定義したファンクションで、列CONTENTに含まれるイメージURLに、APEXセッションで認証する引数**_apex_session**を追加しています。

CKEditorとは異なり、urlconverter_callbackによるイメージURLの置き換えは、ページ・アイテムP2_CONTENTが保持しているデータを置き換えます。そのまま、**作成**や**変更の適用**を実行すると、イメージURLに**?_apex_session=...**の認証子が付加されてデータベースに保存されます。

ワークアラウンドとして、データを保存する前に?_apex_session=...の認証子をP2_CONTENTの内容から取り除くプロセスを作成します。

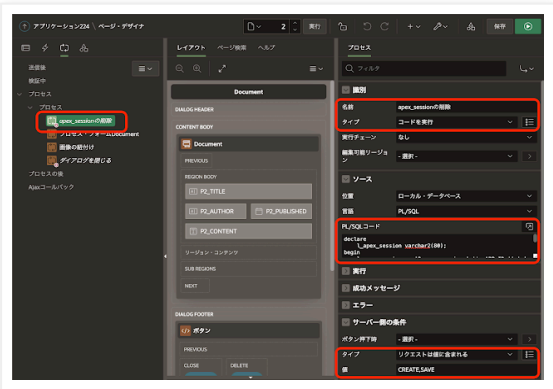
作成したプロセスはプロセス**プロセス・フォームDocument**の上に配置し、先に実行されるようにします。

識別の**名前**は**apex_sessionの削除**、**タイプ**は**コードを実行**、**ソース**の**PL/SQLコード**として以下を記述します。

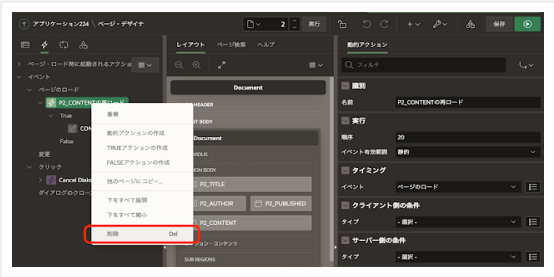
```
declare
    l_apex_session varchar2(80);
begin
    l_apex_session := '?_apex_session=' || :APP_ID || ',' || :APP_SESSION;
    :P2_CONTENT := replace(:P2_CONTENT, l_apex_session, '');
end;
```

remove-apex-session.sql hosted with ❤️ by GitHub　　view raw

サーバー側の条件の**タイプ**に**リクエストは値に含まれる**を選び、**値**に**CREATE,SAVE**を設定します。



CKEditorが必要としていた、ページ・ロード時の動的アクション**P2_CONTENTの再ロード**は、TinyMCEでは不要なので削除します。



以上でTinyMCEへの変更は完了です。アプリケーションはCKEditorのときと同様に動作します。

TinyMCEに変更した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/tinymce-image-sample.zip

オブジェクト・ストレージを使うアプリの変更

オブジェクト・ストレージを使うAPEXアプリは、**?_apex_session=**の認証子をイメージURLに追加する代わりに、オブジェクト・ストレージの事前承認リクエストに置き換えるように変更します。

ページ番号２の**ページ・プロパティ**の**JavaScript**の**ファンクションおよびグローバル変数の宣言**の記述は以下になります。

```
const apexSession = apex.env.APP_ID + ',' + apex.env.APP_SESSION;
const imageUrl = "&G_IMAGE_URL.";

const image_upload_handler = (blobInfo, progress) => new Promise((resolve, reject) => {
    const xhr = new XMLHttpRequest();
    xhr.withCredentials = false;
    xhr.open('POST', imageUrl);
    xhr.setRequestHeader("Apex-Session", apexSession);    // APEXセッションでの認証を付加

    xhr.upload.onprogress = (e) => {
        progress(e.loaded / e.total * 100);
    };

    xhr.onload = () => {
        if (xhr.status === 403) {
            reject({ message: 'HTTP Error: ' + xhr.status, remove: true });
            return;
        }

        if (xhr.status < 200 || xhr.status >= 300) {
            reject('HTTP Error: ' + xhr.status);
            return;
        }
```

```
        const json = JSON.parse(xhr.responseText);

        // json.locationをjson.urlに変更する。
        if (!json || typeof json.url != 'string') {
            reject('Invalid JSON: ' + xhr.responseText);
            return;
        }

        /* 事前承認済みURLに置き換える。 */
        resolve(json.url.replace('&G_OBJECT_STORAGE_URL!RAW.','&G_PREAUTH_URL!RAW.'));
    };

    xhr.onerror = () => {
        reject('Image upload failed due to a XHR Transport error. Code: ' + xhr.status);
    };

    const formData = new FormData();
    formData.append('file', blobInfo.blob(), blobInfo.filename());

    xhr.send(formData);
});
```

tinymce-obs-images-upload-handler.js hosted with ❤️ by GitHub view raw

TinyMCEの**初期化JavaScriptファンクション**の記述は以下になります。

```
function(options){
    // Add the plugin and the toolbar option
    options.editorOptions.plugins += " image";
    options.editorOptions.toolbar += " image";

    // Use the custom upload handler
    options.editorOptions.images_upload_handler = image_upload_handler;

    // Disable relative url
    options.editorOptions.relative_urls = false;

    // Replace the editor content
    options.editorOptions.init_instance_callback = (editor) => {
        editor.setContent(editor.getContent());
    }

    // Define the URL converter and enable it
    options.editorOptions.convert_urls = true;
    options.editorOptions.urlconverter_callback = function(url, node, on_save, name) {
        if (node === "img" && name === "src" ){
            url = url.replace('&G_OBJECT_STORAGE_URL!RAW.','&G_PREAUTH_URL!RAW.');
        }
```

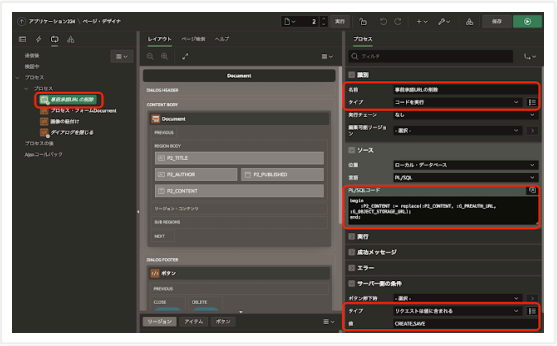
```
        return url;
    }

    return options;
}
```

tinymce-obs-initialize.js hosted with ❤️ by GitHub view raw

ページ・アイテム**P2_CONTENT**を保存する前に、事前承認リクエストを取り除くプロセスのPL/SQLコードは以下になります。**名前**は**事前承認URLの削除**とします。

```
begin
    :P2_CONTENT := replace(:P2_CONTENT, :G_PREAUTH_URL, :G_OBJECT_STORAGE_URL);
end;
```

remove-preuath-url.sql hosted with ❤️ by GitHub view raw



ページ・ロード時の動的アクション**P2_CONTENTの再ロード**を削除すると、TinyMCEの対応は完了です。

アプリケーションのエクスポートは以下に置きました。
https://github.com/ujnak/apexapps/blob/master/exports/tinymce-image-sample-obs.zip

こちらの記事で紹介している動的コンテンツの追加手順に変更はありません。

動的コンテンツのページを追加した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/tinymce-image-sample-dyn.zip

以上になります。

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: 15:27

共有

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