

WebClapModoki_GAS

(google スプレッドシート & Google Apps Script(GAS) で Web拍手っぽい何かを実装してみた)

処理内容・導入手順の詳細説明スライド

2022/12/25 初版

文責: dullNeko

GitHub: https://github.com/dullNeko/WebClapModoki_GAS

Twitter: @dullNeko

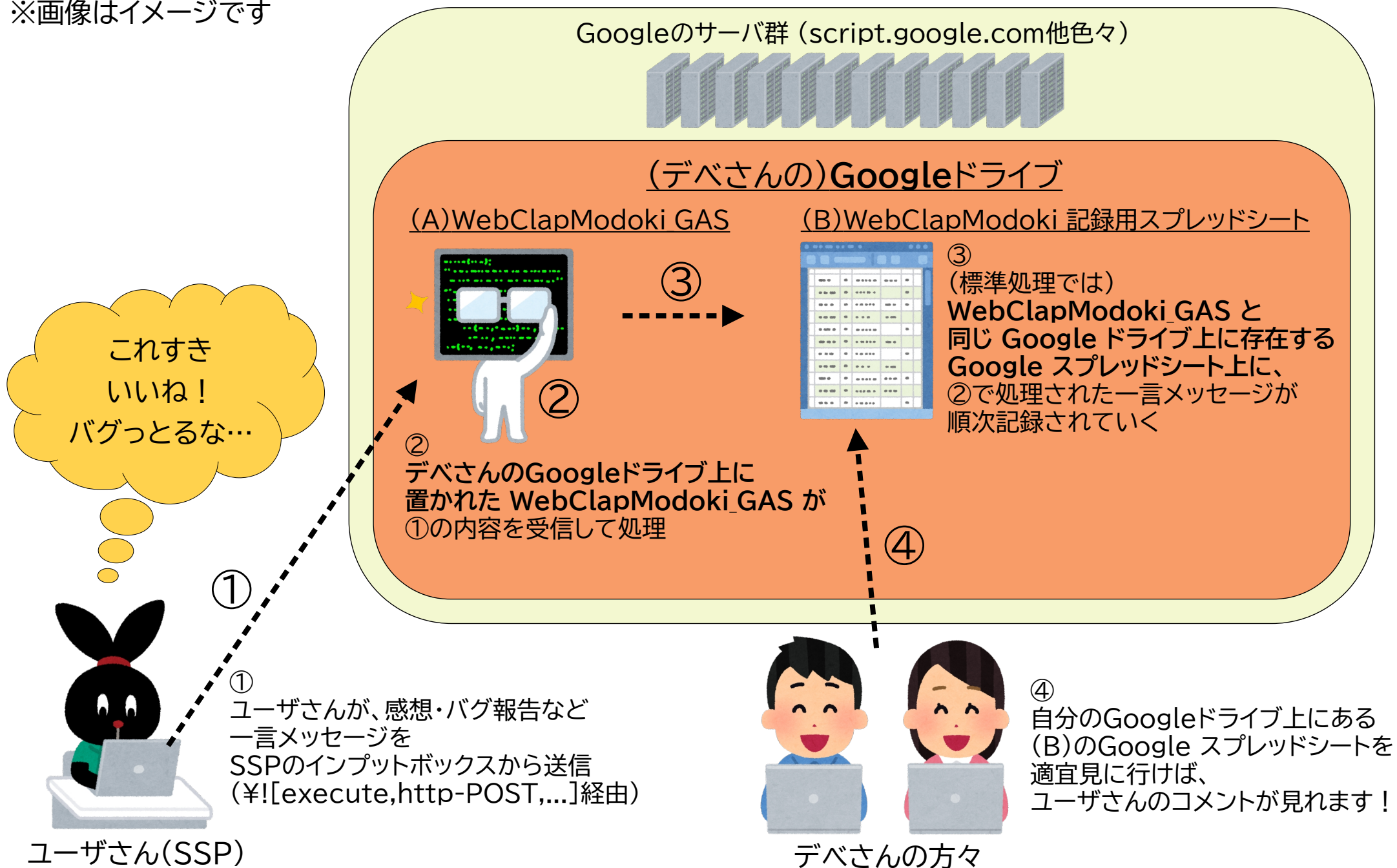
Mastodon: @dullNeko@ukadon.shillest.net

●目次

- (1) 【処理の図解】これ、どんな処理するの？
- (2) 【準備するもの】必要なものは何？
- (3) 【環境作成・設定方法】どうすれば動くの？
 - (3.1) Google ドライブ上に必要なファイルを用意
 - (3.2) Google Apps Script の準備／動作確認／設定変更
 - (3.3) Google Apps Script のデプロイ／WebアプリURL取得
 - (3.4) SSP側でのHTTPリクエストの実装／動作テスト

(1)【処理の図解】これ、どんな処理するの？

※画像はイメージです



(2)【準備するもの】必要なものは何？

以下の【1】【2】【3】【4】が必須です。

【2】【3】【4】については、
適切な場所への設置・設定書き換え等を行う必要があります。

- 【1】 Google アカウント
+ Google ドライブの空き容量(数MBくらい)
- 【2】 dullNeko の Google ドライブ上で公開されている WebClapModoki (template) のコピー
https://docs.google.com/spreadsheets/d/1j_h4weHihW-uVeTFtHTwrs6QwRPAT36sEBf3DPbo3ms/edit#gid=0
- 【3】 dullNeko の GitHub リポジトリにある WebClapModoki_GAS.txt の中身
<https://raw.githubusercontent.com/dullNeko/WebClapModoki-GAS/main/WebClapModoki-GAS.txt>
- 【4】 デベさんのゴーストさん内での `\\[execute,http-POST,...]` の記述
 - ※ POSTメソッドを使ってHTTPリクエストを送れるモノなら、送信側はSSPに限りません：
例えばご自分のWebサイトに設置したスクリプトとか、あるいは自作ツールに実装したバグ報告機能とか、もOK(なはず)です。

(3) 【環境作成・設定方法】どうすれば動くの？

大きく分けて、次の4段階で設定していきます。

- (3.1) Google ドライブ上に必要なファイルを用意
- (3.2) Google Apps Script の準備／動作確認／設定変更
- (3.3) Google Apps Script のデプロイ／WebアプリURL取得
- (3.4) SSP側でのHTTPリクエストの実装／動作テスト

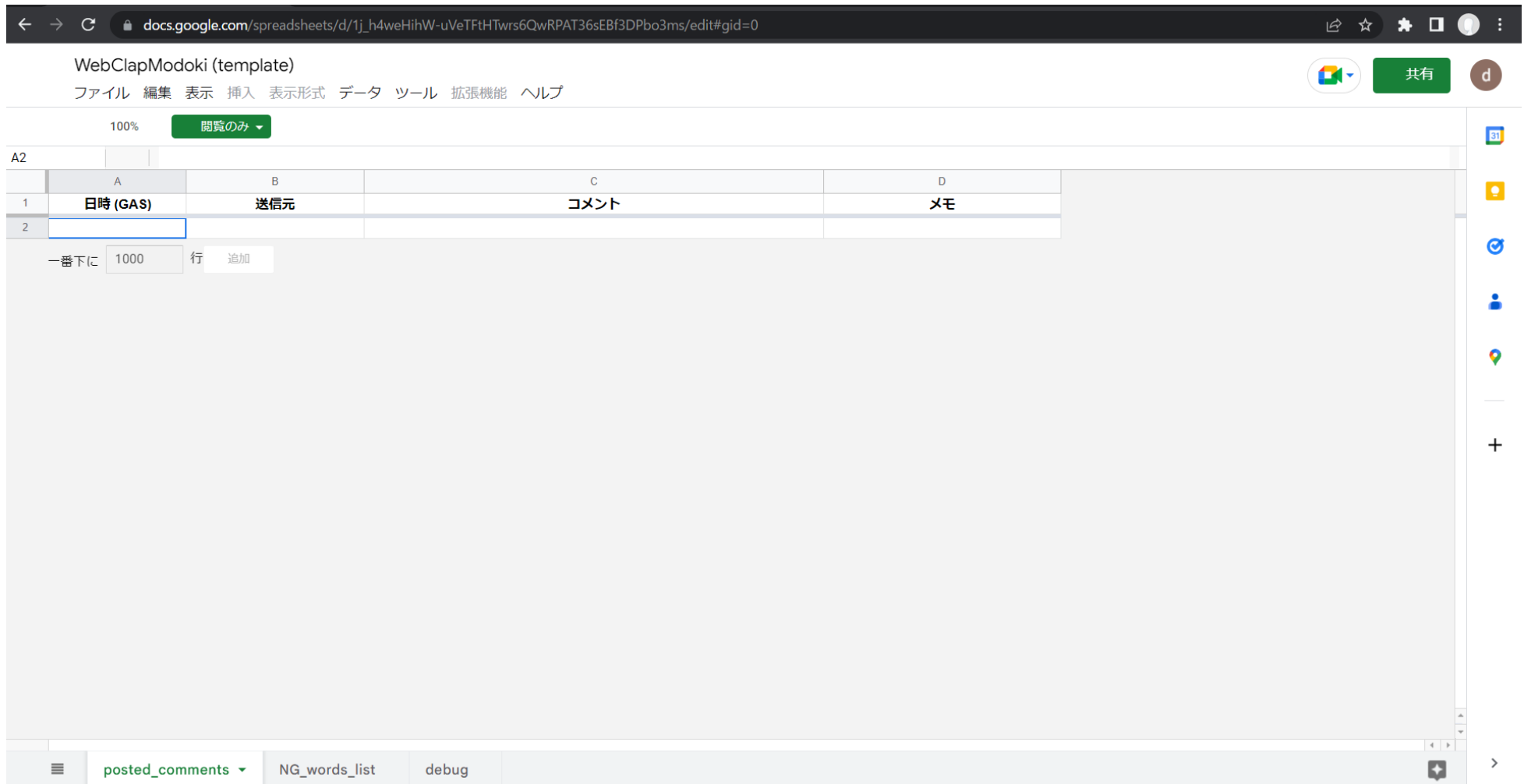
次スライドから、順番に説明していきますね。

(3.1) Google ドライブ上に必要なファイルを用意 (1/23)

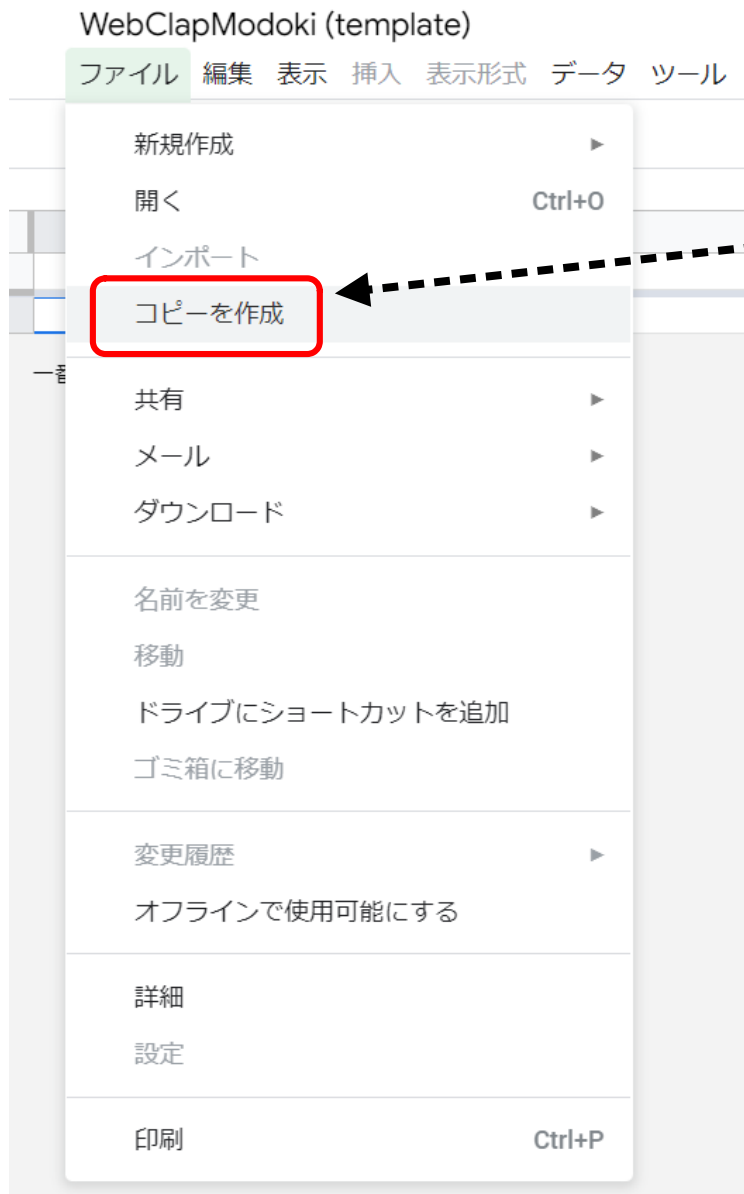
- (1) まず、**Googleアカウントでログインした状態のブラウザ上で**
dullNeko の Google ドライブ上で公開されている、次のスプレッドシートを開いて下さい。

https://docs.google.com/spreadsheets/d/1j_h4weHihW-uVeTFtHTwrs6QwRPAT36sEBf3DPbo3ms/edit?usp=sharing

…次の画面が出てきたでしょうか？



(3.1) Google ドライブ上に必要なファイルを用意 (2/23)



(2)
メニューにある
「ファイル」→「コピーを作成」
を開きます。

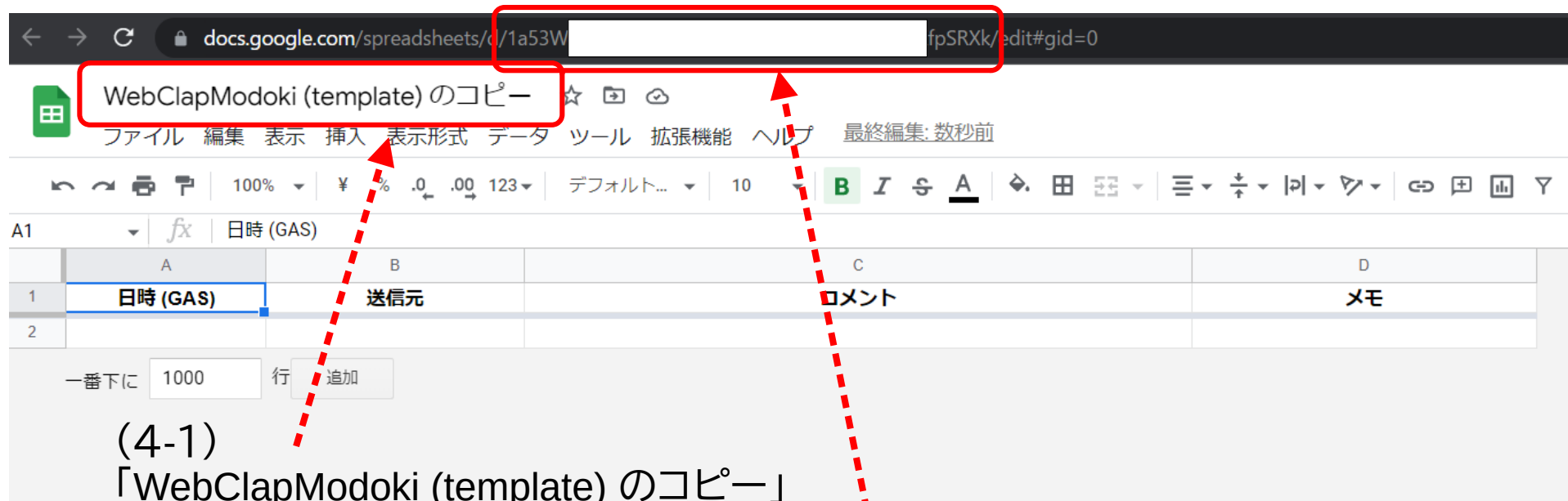
※ 「コピーを作成」が
グレースアウトして選択できない場合は、
「Googleアカウントのログイン」を
やり直してみてください。

(3.1) Google ドライブ上に必要なファイルを用意 (3/23)



(3)
「コピーを作成」をクリックして実行します。

(3.1) Google ドライブ上に必要なファイルを用意 (4/23)



(4-1)
「WebClapModoki (template) のコピー」
が作成できればOK！

(4-2)
※重要ポイント※
この白抜きで一部隠してある部分が
「(スプレッドシートの) id」と呼ばれる、
「スプレッドシートを特定する」大変重要な情報です。

[https://docs.google.com/spreadsheets/d/
1a53W*****fpSRXk/edit#gid=0](https://docs.google.com/spreadsheets/d/1a53W*****fpSRXk/edit#gid=0)

後の工程で必要になる情報なので、
控えておくとスムーズです。
同時に、人に教えてはいけない機密情報でもあります。
取り扱い注意です！

(3.1) Google ドライブ上に必要なファイルを用意 (5/23)



(5)
さて、Google ドライブの画面に戻ってみましょう。
ちゃんと「WebClapModoki (template)」のコピーが
表示されていればOKです！

※ 本手順書では変えていませんが、先の id でシートを特定しているため、
このファイルの名前は自由に変えて大丈夫です。
「ゴーストの感想」など、ご自身が分かりやすいようにリネームして下さい。

(3.1) Google ドライブ上に必要なファイルを用意 (6/23)

マイドライブ ▾

ファイル

WebClapModoki (temp...

新しいフォルダ

ファイルのアップロード

フォルダのアップロード

Google ドキュメント >

Google スプレッドシート >

Google スライド >

Google フォーム >

その他 >

(6)

これで (2) 【準備するもの】で言及していた
【2】スプレッドシートのコピーが用意できました！

お次は、

【3】の Google Apps Script ファイルを用意します。

Google ドライブの、
背景の何も無いところを右クリックすると、
←のメニューが出てきますので、
「その他」→「Google Apps Script」を選んで
クリックしましょう。

Google 図形描画

Google マイマップ

Google サイト

Google Apps Script

Google Jamboard

+ アプリを追加

(3.1) Google ドライブ上に必要なファイルを用意 (7/23)



(7)
「Apps Script - 無題のプロジェクト」という画面が、
新しいタブで開いたのでしょうか？
この画面が出てきたら、ひとまずOKです。
次のスライドへどうぞ。

(3.1) Google ドライブ上に必要なファイルを用意 (8/23)

(8)
一旦、dullNeko の GitHub リポジトリ上にある WebClapModoki_GAS.txt を開いて下さい。
RAW(テキスト)表示のURL(↓)を開き、Ctrl+A で全選択してから Ctrl+C でコピーして、

<https://raw.githubusercontent.com/dullNeko/WebClapModoki-GAS/main/WebClapModoki-GAS.txt>

先の Apps Script のエディタに Ctrl+V で 全文を 貼り付けて、Ctrl+S で上書き保存して下さい。

```
//=====
// WebClapModoki_GAS_v7
//
// 【作成者】
// dullNeko (https://github.com/dullNeko/WebClapModoki_GAS)
//
// 【更新履歴】
// 2022/12/21 v7 : 公開用に調整した最初のバージョン。
//                MastodonへのAPI経由でのツイート機能を試験実装するとともに、
//                セーフティとしてプロパティサービスを利用しないと動作しないように調整。
// 2022/12/15 v1 : Web拍手代替物がGASで組めるかのテスト版。
//                とりあえず動作確認できたレベル。
//=====

// ★ 参考資料
//=====
/*
覚えておいた方がよいページなど。
「直接コードを引用した」「大いに参考にさせて頂いたページ」等は
コード周辺に配置して明示するようにしています。
*/

//=====
// ★ GAS全体の制限について
//=====
/*
■ Google サービスの割り当て | Apps Script | Google Developers
https://developers.google.com/apps-script/guides/services/quotas

GASは基本的に無償で利用できるサービスですが、
代わりに、上記URLのページの通り「実行可能量」の制限があります。
> Apps Script サービスには、1日あたりの割り当てと、一部の機能に対する制限があります。
> 割り当てまたは上限を超えると、スクリプトが例外をスローし、実行が停止します。
*/
```

全文コピペ

```
1094 debugPrintReturnCode("500");
1095 debugPrintProgressCheck("ERROR / comment contains NG word.");
1096 return;
1097
1098 */
1099
1100 //=====
1101 // ◆ 得られた各パラメータと、GAS側の日時を合わせてシート (SHEET_COMMENT) に記録する処理
1102 //=====
1103
1104 // 時系列順にソートしやすくするため、
1105 // GAS 側の DateTime を取得し、これを先頭列に入れます。
1106 let GAS_Date_Time = new Date();
1107
1108 // ここまでの処理で得られた値を追記。
1109 // 行を一括追加する場合、setValuesメソッドよりappendRowメソッドの方が楽でした。
1110 SHEET_COMMENT.appendRow([GAS_Date_Time, sender, comment, memo]);
1111
1112 //=====
1113 // ◆ ついでに Mastodon にツイートしてみる処理
1114 // ※実験的機能です。
1115 // tootByMastodonAPI関数の中身を良く確認し、十分注意してご使用下さい。
1116 //=====
1117
1118 let res_mastodon = tootByMastodonAPI("sender: " + sender + "\r\n" + "comment: " + comment + "\r\n" + "memo: " + memo);
1119 debugPrint("responce from mastodon API", res_mastodon);
1120
1121 //=====
1122 // ◆ 終了処理
1123 //=====
1124
1125 debugPrintProgressCheck("step final / normal end");
1126 debugPrintReturnCode("200");
1127 return;
1128
1129
1130
```

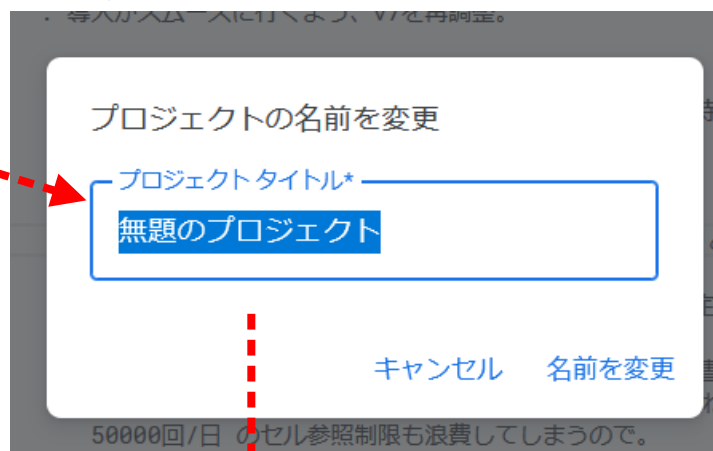
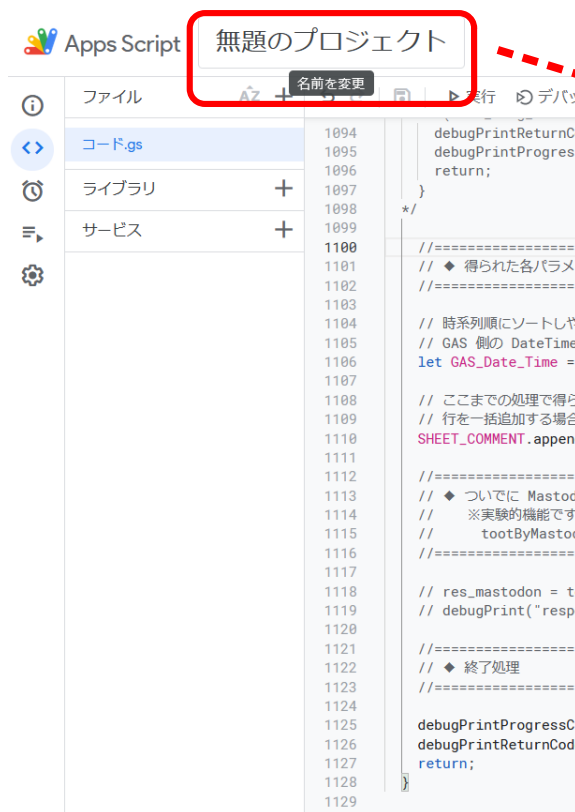
(3.1) Google ドライブ上に必要なファイルを用意 (9/23)

(9)

タイトルが「無題のプロジェクト」のままでも、動作に支障はない、のですが…後々、混乱のもとになりやすいので、ここで変えておきましょう。

(10-1)

「無題のプロジェクト」付近にカーソルを持っていくと、「名前を変更」という表示が出てくるので、そこで左クリック。



(10-2)

「プロジェクトの名前を変更」ウィンドウが出てくるので、「WebClapModoki_GAS_v8」あたりに変えます。



(10-3)

変え終わったら、「名前を変更」をクリックして、確定させて下さい。

(3.1) Google ドライブ上に必要なファイルを用意 (10/23)

(10)

元の画面に戻って、タイトルが「WebClapModoki_GAS_v8」に変更されていればOKです。

それでは、(4-2)で控えた

「**スプレッドシートの id**」を、本スクリプトに記入していく、GASの準備段階に移ります。
次のスライドへどうぞ。

Apps Script WebClapModoki_GAS_v8 デプロイ

ファイル AZ + 実行 デバッグ checkPropertyValue 実行ログ

コード.gs

```
1 //=====
2 // WebClapModoki_GAS_v8
3 //
4 // 【作成者】
5 // dullNeko (https://github.com/dullNeko/WebClapModoki\_GAS)
6 //
7 // 【更新履歴】
8 // △ ... 機能追加など
9 // ※ ... コードの整理・修正・バグ対応など
10 // ▼ ... 機能の削除など
11 //
12 // 2022/12/24 v8 : 導入がスムーズに行くよう、v7を再調整。
13 //
14 // ※初期設定を簡単にするため、
15 // Googleスプレッドシート（のテンプレート）のコピーを持ってきて、
16 // その 'id' をこのスクリプト内の
17 // const SPREADSHEET_ID = 'XXX...XXX';
18 // に入れば、
19 // 即 doPostTest() 関数でのテストを通過し、Webアプリとしてのデプロイまでできるように整理。
20 //
21 // ※その一環として、 const DEBUG_DISABLE = 1; を既定値に変更。
22 // 問題があった時に使うだけで十分だし、
23 // DEBUG_DISABLE = 1 (=デバック関数によるセル参照・書込を禁止)にしていないと、
```

(3.2) Google Apps Script の準備／動作確認／設定変更 (11／23)

(11)

(4-2)の「**スプレッドシートの id**」を、本スクリプトの 325行目の 'XXX...XXX' に入れて、

const SPREADSHEET_ID = '**スプレッドシートの id**';

として書き換えます。

書き換えたら、Ctrl+Sで上書き保存を忘れずに。

(※先述した通り、機密情報なので一部黒塗りです)

Apps Script WebClapModoki_GAS_v8

```
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
```

コード.gs

ライブラリ

サービス

実行 デバッグ checkPropertyValue 実行ログ

なお、スプレッドシート自体、ならびに本スクリプトの共有設定は「制限 (=スプレッドシート、このGAS本体(コード.gs)を公開する必要はありません) 」。ただし、本スクリプトのデプロイ時の「アクセスできるユーザー」を「全発行されるWebアプリケーションのURL https://script.google.com/macros/s/YYYY...YYY/exec については、辞書ファイル内に記述する (=ユーザーさんに公開する) 必要

【！注意！】
この 'id' は、
「あなたのゴーストさんの感想が詰まったスプレッドシート」を特定でき
基本的に自分以外の誰か (ユーザーさん、他のデベさん、友人知人家族) に
取り扱いには十分ご注意下さい。

const SPREADSHEET_ID = 'XXX...XXX';

// =====
// ● SHEET_NAME_COMMENT
// =====
/*
↑の 'id' で特定したスプレッドシート内の、コメントを記録するシート
私は 'posted_comments' で固定しています。
*/
const SHEET_NAME_COMMENT = 'posted_comments';
// =====
// ● SHEET_NAME_NGWL
// =====
/*
↑の 'id' で特定したスプレッドシート内の、
ブラックリスト式のNGワードリストを 'シート名' で指定する用。
私は 'NG_words_list' で固定しています。
*/
const SHEET_NAME_NGWL = 'NG_words_list';
// =====
// ● SHEET_NAME_DEBUG
// =====

lapModoki_GAS_v8

実行 デバッグ checkPropertyValue 実行ログ

311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347

なお、スプレッドシート自体、ならびに本スクリプトの共有設定は「制限付き」でOKです。
(=スプレッドシート、このGAS本体(コード.gs)を公開する必要はありません)
ただし、本スクリプトのデプロイ時の「アクセスできるユーザー」を「全員」にした上で、
発行されるWebアプリケーションのURL
https://script.google.com/macros/s/YYYY...YYY/exec
については、辞書ファイル内に記述する (=ユーザーさんに公開する) 必要があります。

【！注意！】
この 'id' は、
「あなたのゴーストさんの感想が詰まったスプレッドシート」を特定できてしまう情報なので、
基本的に自分以外の誰か (ユーザーさん、他のデベさん、友人知人家族) に公開してはいけません。
取り扱いには十分ご注意下さい。

const SPREADSHEET_ID = '1pC [redacted] dprtM';

// =====
// ● SHEET_NAME_COMMENT
// =====
/*
↑の 'id' で特定したスプレッドシート内の、コメントを記録するシートを 'シート名' で指定
私は 'posted_comments' で固定しています。
*/
const SHEET_NAME_COMMENT = 'posted_comments';
// =====
// ● SHEET_NAME_NGWL
// =====
/*
↑の 'id' で特定したスプレッドシート内の、
ブラックリスト式のNGワードリストを 'シート名' で指定する用。
私は 'NG_words_list' で固定しています。
*/
const SHEET_NAME_NGWL = 'NG_words_list';
// =====
// ● SHEET_NAME_DEBUG
// =====

(3.2) Google Apps Script の準備／動作確認／設定変更 (12／23)

(12)

これで準備が整いました。

それでは、動作確認をしていきましょう。

「デバッグ」の右にある「関数」のリストから、「doPostTest」を選択します。

Apps Script WebClapModoki_GAS_v8

①この枠内をクリックすると、「関数」のリストが展開されます。

The screenshot shows the Google Apps Script editor interface. On the left, the 'Functions' dropdown menu is open, displaying a list of functions: `checkPropertyValue`, `checkPropertyValue_API_KEY_MASTODON`, `debugPrint`, `debugPrintProgressCheck`, `debugPrintReturnCode`, `doPostTest` (highlighted with a red box), `listupArray`, `checkContentTextByNGWordsList`, and `tootByMastodonAPI`. A red dashed arrow points from the 'checkPropertyValue' dropdown to the 'doPostTest' function. The background code is as follows:

```
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
```

Comments in the code include: 「なお、スプレッドシート自体、 (=スプレッドシート、このG」、 「ただし、本スクリプトのデプロ」、 「発行されるWebアプリケーション」、 「https://script.google.c」、 「については、辞書ファイル内に」、 「【！注意！】」、 「ここの 'id' は、」、 「「あなたのゴーストさんの感」、 「基本的に自分以外の誰か (ユー」、 「取り扱いには十分ご注意下さい」、 「↑の 'id' で特定したスプレッドシート内の、 コメントを記録するシートを」、 「私は 'posted_comments' で固定しています。」、 「↑の 'id' で特定したスプレッドシート内の、」、 「ブラックリスト式のNGワードリストを 'シート名' で指定する用。」、 「私は 'NG_words_list' で固定しています。」

②出てきたリスト内の、「doPostTest」をクリックして、選択します。

(3.2) Google Apps Script の準備／動作確認／設定変更 (13／23)

(13)

選択された関数が「doPostTest」になっていることを確認して、「実行」ボタンをクリックします。



Apps Script WebClapModoki_GAS_v8

ファイル **▶ 実行** デバッグ doPostTest ▼ 実行ログ

コード.gs

ライブラリ +

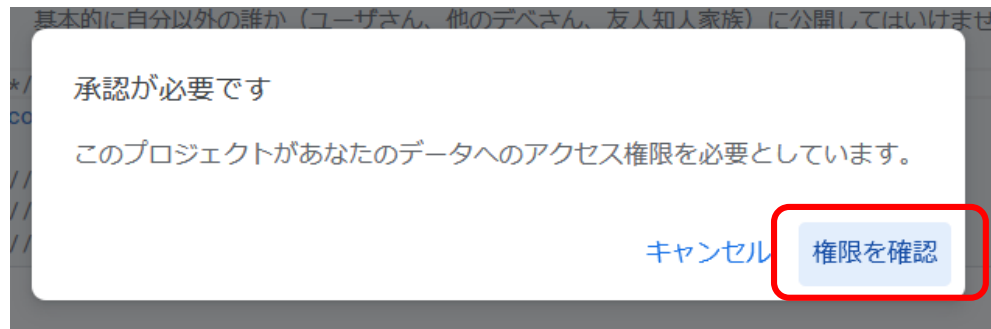
サービス +

311
312 なお、スプレッドシート自体、ならびに本スクリプトの共有設定は
313 (=スプレッドシート、このGAS本体(コード.gs)を公開する必要
314 ただし、本スクリプトのデプロイ時の「アクセスできるユーザー」
315 発行されるWebアプリケーションのURL
316 <https://script.google.com/macros/s/YYYY...YYY/exec>
317 については、辞書ファイル内に記述する (=ユーザーさんに公開する
318
319 **【！注意！】**
320 ここの 'id' は、
321 「あなたのゴーストさんの感想が詰まったスプレッドシート」を特

(3.2) Google Apps Script の準備／動作確認／設定変更 (14／23)

(14)
…ここで、記事本文の【注意喚起】で記述していた、
「承認が必要です:このプロジェクトがあなたのデータへのアクセス権限を必要としています」
ウィンドウが出てきます。

以降、「アクセス権を与えて、実行を許可する」方法を示します。



(14-1)
「権限を確認」ボタンをクリックします。



(14-2)
(この例では私のアカウントですが)
ご自身のアカウントが表示されるので、
それをクリックして選択します。

(3.2) Google Apps Script の準備／動作確認／設定変更 (15／23)

(15)

個人が作成したスクリプトなので、
下記の警告(Googleのチェックを受けていないが良いか?)が出ます…が、
許可しないと実行できないので、先に進みます。



このアプリは Google で確認されていません

アプリが、Google アカウントのプライベートな情報へのアクセスを求めています。デベロッパー ([redacted]@gmail.com) と Google によって確認されるまで、このアプリを使用しないでください。

詳細

安全なページに戻る



このアプリは Google で確認されていません

アプリが、Google アカウントのプライベートな情報へのアクセスを求めています。デベロッパー ([redacted]@gmail.com) と Google によって確認されるまで、このアプリを使用しないでください。

詳細を非表示

安全なページに戻る

(15-1)

「権限を確認」ボタンをクリックします。

リスクを理解し、デベロッパー ([redacted]@gmail.com) を信頼できる場合のみ、続行してください。

無題のプロジェクト (安全ではないページ) に移動

(15-2)

「無題のプロジェクト(安全ではないページ)に移動」
ボタンをクリックします。

(3.2) Google Apps Script の準備／動作確認／設定変更 (16／23)

(16)

最終確認画面です。

「許可」をクリックすれば、以下のアクセス権限が与えられて、スクリプトを実行可能な状態になります。



(16-1)

「許可」ボタンをクリックします。

(3.2) Google Apps Script の準備／動作確認／設定変更 (17／23)

(17)
(16)で「許可」をクリックすると、元の画面に自動で戻り、doPostTest関数の実行が行われます。
ここまでで設定ミス等がなければ、
「実行開始」の表示後、数秒内に、「お知らせ 実行完了」表示が出ます。
これで、正常に動作することが確認できたので、次にデプロイを行います。



The screenshot shows the Google Apps Script editor interface. The top bar displays "Apps Script" and the project name "WebClapModoki_GAS_v8". A blue "デプロイ" (Deploy) button is visible. The left sidebar shows the file explorer with "コード.gs" (Code.gs) selected. The main editor area shows the code for the doPostTest function. The execution log at the bottom shows two messages: "18:17:47 お知らせ 実行開始" (18:17:47 Notice Execution Start) and "18:17:49 お知らせ 実行完了" (18:17:49 Notice Execution Complete). The second message is highlighted with a red box.

```
1 //=====
2 // WebClapModoki_GAS_v8
3 //
4 // 【作成者】
5 // dullNeko (https://github.com/dullNeko/WebClapModoki\_GAS)
6 //
7 // 【更新履歴】
8 // △ ... 機能追加など
9 // ※ ... コードの整理・修正・バグ対応など
10 // ▼ ... 機能の削除など
11 //
12 // 2022/12/24 v8 : 導入がスムーズに行くよう、v7を再調整。
13 //
14 // ※初期設定を簡単にするため、
15 // Googleスプレッドシート (のテンプレート) のコピーを持ってきて、
16 // その 'id' をこのスクリプト内の
17 // const SPREADSHEET_ID = 'XXX...XXX';
18 // に入れれば
```

実行ログ

18:17:47	お知らせ	実行開始
18:17:49	お知らせ	実行完了

(17-1)
「お知らせ 実行完了」が表示されていればOKです。

(3.3) Google Apps Script のデプロイ／WebアプリURL取得 (18／23)

(18)
「デプロイ」ボタンをクリックして、
その中の「新しいデプロイ」をクリックします。

The screenshot shows the Google Apps Script editor for a project named 'WebClapModoki_GAS_v8'. The left sidebar contains a file explorer with 'コード.gs' (code.gs) selected. The main editor area displays the code for 'doPostTest'. The code includes comments in Japanese and a constant for 'SPREADSHEET_ID'. The right sidebar shows the 'Deploy' button, which is highlighted with a red box. A red dashed arrow points from this button to a zoomed-in view of the deployment menu. In the menu, the '新しいデプロイ' (New deployment) option is highlighted with a red box. Below the menu, the execution log shows a successful deployment at 18:17:49.

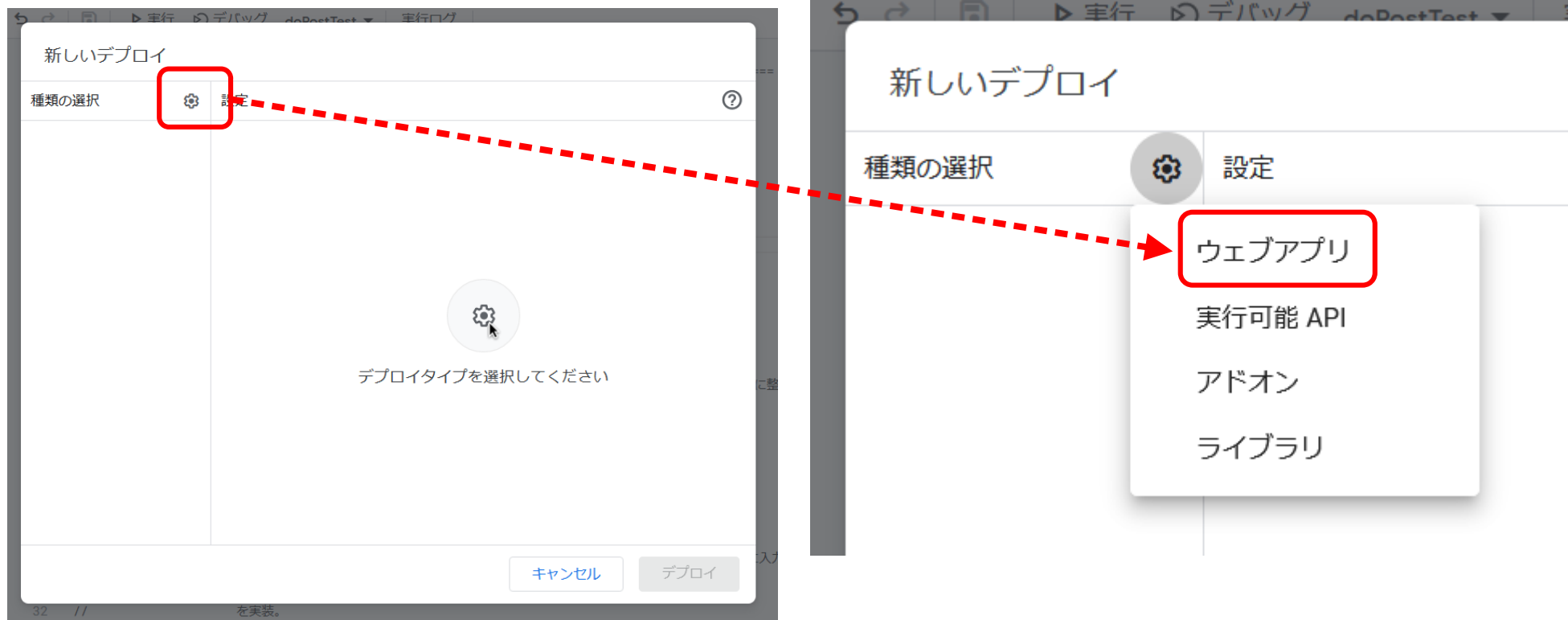
```
1 //=====
2 // WebClapModoki_GAS_v8
3 //
4 // 【作成者】
5 // dullNeko (https://github.com/dullNeko/WebClapModoki_GAS)
6 //
7 // 【更新履歴】
8 // △ ... 機能追加など
9 // ※ ... コードの整理・修正・バグ対応など
10 // ▼ ... 機能の削除など
11 //
12 // 2022/12/24 v8 : 導入がスムーズに行くよう、v7を再調整。
13 //
14 // ※初期設定を簡単にするため、
15 // Googleスプレッドシート（のテンプレート）のコピーを持ってきて、
16 // その 'id' をこのスクリプト内の
17 // const SPREADSHEET_ID = 'XXX...XXX';
18 // に入れれば
```

実行ログ

18:17:47	お知らせ	実行開始
18:17:49	お知らせ	実行完了

(3.3) Google Apps Script のデプロイ／WebアプリURL取得 (19／23)

(19)
「種類の選択」の右にある「歯車」ボタンをクリックし、
その中の「ウェブアプリ」をクリックします。



(3.3) Google Apps Script のデプロイ／WebアプリURL取得 (20／23)

(20)
「アクセスできるユーザー」をクリックし、
その中の「全員」をクリックして設定します。

新しいデプロイ

種類の選択 設定

ウェブアプリ

説明

新しい説明文

ウェブアプリ

次のユーザーとして実行: 自分 (illyghok384@gmail.com)

このウェブ アプリケーションを実行するために、あなたのアカウント データを使用することを許可します。

アクセスできるユーザー 自分のみ

ライブラリとしても利用できます。詳細

キャンセル デプロイ

ウェブアプリ

次のユーザーとして実行: 自分 (illyghok384@gmail.com)

このウェブ アプリケーションを実行するために、あなたのアカウント データを使用することを許可します。

アクセスできるユーザー

自分のみ

自分のみ

Google アカウントを持つ全員

全員

キャンセル デプロイ

を実装。

(3.3) Google Apps Script のデプロイ／WebアプリURL取得 (21／23)

(21)

「デプロイ」をクリックして実行し、
Webアプリとして使用する用のURLを発行します。
「デプロイを更新しました。」が表示されたら、
「ウェブアプリ」のURLを「コピー」し、「完了」ボタンをクリックして閉じて下さい。

The screenshot shows the '新しいデプロイ' (New Deployment) dialog box. It has two tabs: '種類の選択' (Select Type) and '設定' (Settings). The '種類の選択' tab is active, showing 'ウェブアプリ' (Web App) as the selected type. Below this, there is a '説明' (Description) field with the placeholder '新しい説明文'. Under the 'ウェブアプリ' section, there are two dropdown menus: '次のユーザーとして実行:' (Run as the following user:) set to '自分 (illyghok384@gmail.com)' and 'アクセスできるユーザー' (Users who can access) set to '全員' (Everyone). A blue button 'デプロイ' (Deploy) is at the bottom right, highlighted with a red box. A red dashed arrow points from this button to the '新しいデプロイ' dialog box on the right.

The screenshot shows the '新しいデプロイ' (New Deployment) dialog box after deployment. It displays the message 'デプロイを更新しました。' (Deployment updated). Below this, it shows 'バージョン 1 (12月24日 21:47)' (Version 1 (December 24, 21:47)) and the 'デプロイ ID' (Deployment ID) 'AKfycbx-d5QAA3DRcBuSNFMMAp1HXrF7kAtCE1D4nn4eVQK-5wllrDIAxTHwFuBwlaubW-EZkg'. There is a 'コピー' (Copy) button next to the ID. The 'ウェブアプリ' (Web App) section shows the 'URL' 'https://script.google.com/macros/s/AKfyc...'. There is a 'コピー' (Copy) button next to the URL, highlighted with a red box. A red dashed arrow points from this button to the '完了' (Done) button at the bottom right, which is also highlighted with a red box.

(3.4) SSP側でのHTTPリクエストの実装／動作テスト (22／23)

(22)

次の `\\[execute,http-POST,(WCM_URL),...]` 文の(WCM_URL)部分に、
(21)でコピーしたWebアプリ用URLを代入し、
ゴーストさんの辞書内に配置して、呼び出してください。

```
\\[execute,http-POST,(WCM_URL),--param=message_body="(送信元);(送信内容);(メモ)",--async=OnWebClapModoki,--timeout=5]
```

里々であれば、本プロジェクト内の dic_wcm.txt をフォルダに入れ、
https://github.com/dullNeko/WebClapModoki-GAS/blob/main/dic_wcm.txt
\$WCM_URL <https://script.google.com/macros/s/YYYY~YYY/exec>
のURLを、(21)でコピーしたWebアプリ用URLに置き換えて上書き保存し、
* WebClapModoki を呼び出せばOKです。(送信元)、(メモ)は、お好みで変更してください。



```
16 # *OnUserInputCancel
17 # > (R 0) を空打ちされた (compare, (R 1),close)
18
19 *WebClapModoki
20 \\[open,inputbox,WEB拍手もどき入力,--timeout=-1,--text=いいね!,--limit=800]
21 \\p[0] (作者にメッセージを送ることができます)
22 (感想・要望・バグ報告など、お気軽にどうぞ！)
23 # 一応、上限値として --limit=800 を指定していますが、これは厳密にテストした値ではありません。
24 # GASのスク립ト実行時間制限 (6分/回)、
25 # SSPのインプットボックスで表示できる領域の大きさ…等を鑑みると、
26 # あまり上限値を大きくしても意味がなさそう、と判断した結果です。
27
28 # 【参考】
29 # ■ UKADOC・Project さくらスク립トリスト
30 # https://ssp.shillest.net/ukadoc/manual/list_sakura_script.html
31 # \\[open,inputbox,ID,表示時間,テキスト,オプション,...]
32
33 *WEB拍手もどき入力をおこなった
34 $WCM_URL https://script.google.com/macros/s/AKfyd...EZkg/exec
35 $送信元 Net:Withered Lily
36 $送信内容 (R 1)
37 $メモ メインメニューより送信されたコメント
38 \\p[0] (拍手送信開始...)
39 \\[execute,http-POST,(WCM_URL),--param=message_body="(送信元);(送信内容);(メモ)",--async=OnWebClapModoki,--timeout=5]
40
41 # 【※注意※】
42 # $WCM_URL の内容を「ご自身でデプロイして取得したWebアプリのURL」に書き換えないと動きません。ご注意くださいまし。
43 # OnUserInput イベントを経由しているため、
44 # (R 1) = (Reference1) が「先の inputbox で入力された内容」です。
```

(3.4) SSP側でのHTTPリクエストの実装／動作テスト (23／23)

(23)

実際にゴーストさんから呼び出してみて、
記事冒頭のGIF画像のように、
記録先スプレッドシートに記録されることが確認できたら、
初期設定完了です！お疲れ様でした。

WebClapModoki-GAS ☆ 📁 🌐

ファイル 編集 表示 挿入 表示形式 データ ツール 拡張機能 ヘルプ 最終編集: 5分前 (dull neko さん)

100% 123 10 B I A

C3 fx いいね！

	A	B	C	D
1	日時 (GAS)	送信元	コメント	メモ
2	2022/12/24 22:06:57	Not_Withered_Lily	いいね！	メインメニューより送信されたコメント
3	2022/12/24 22:07:07	Not_Withered_Lily	いいね！	メインメニューより送信されたコメント

一番下に 1000 行 追加

+ posted_comments NG_words_list debug

