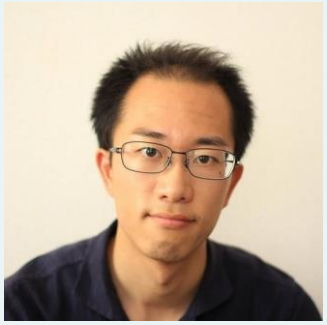
Google Apps Scriptによる
快適なデータ操作を考えてみた

2019年 11月

那須野 拓実



那須野 拓実 （なすの たくみ）

https://takuminasuno.com/ja/

たなぐら応援大使（福島県棚倉町）。
トリプレッソを勝手に応援する人。
ネイチャーフォト中心の多言語ブログを書いてます。

本業はIT&マーケティング界隈で
ナレッジマネジメントとかデータ分析とかの何でも屋。
半年間の育休明けで、家事育児と外働きのバランスを模索中。

コンタクトはこちらからどうぞ！ ⇒ takumi_nasuno

※上記の自己紹介は、2019年11月執筆当時のものです。

GASを使うとき
データをどこに保存するべきか

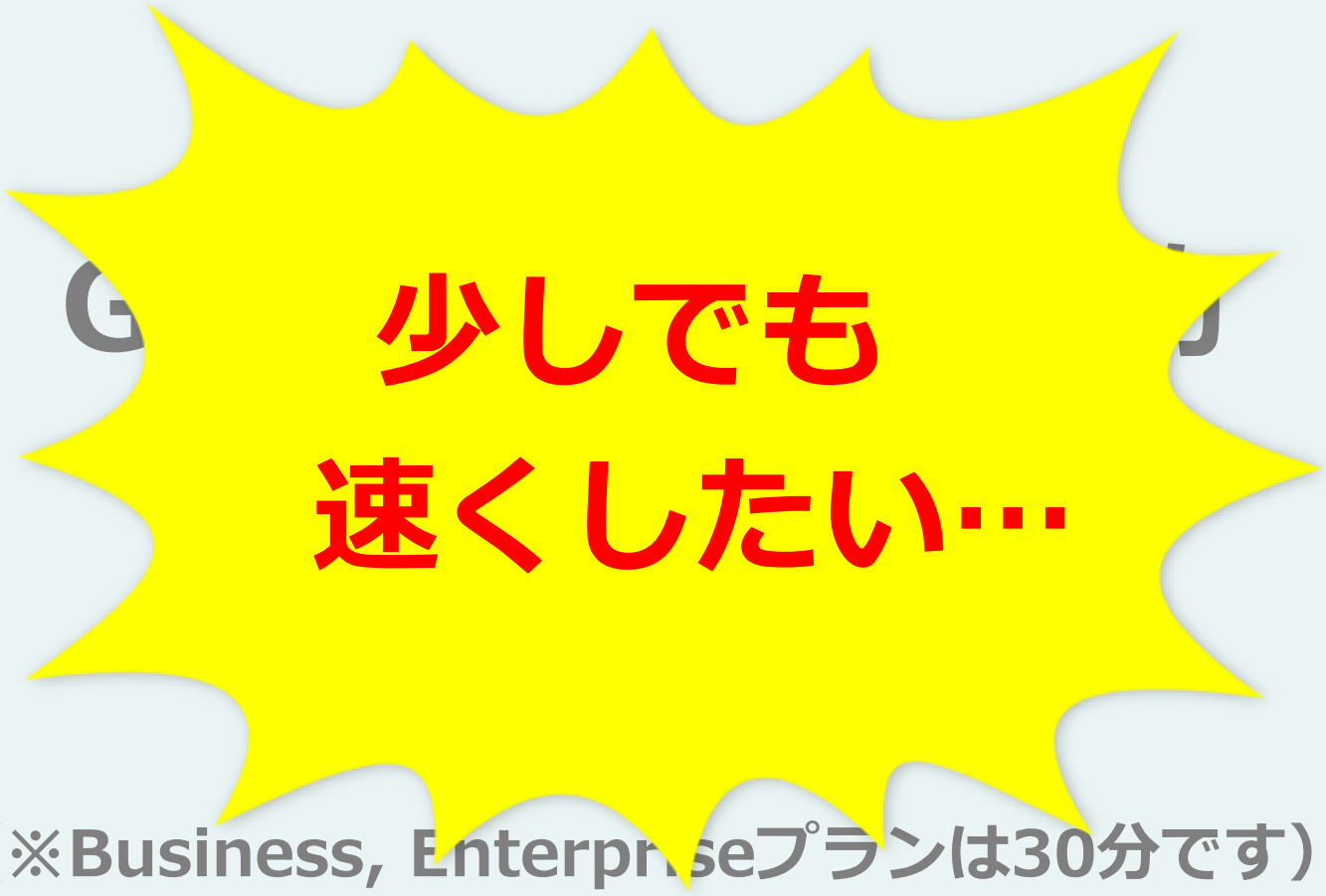
本当にスプレッドシートで良いのだろうか？

GASの気になる制約

GASの気になる制約

起動時間6分の壁

（※Business, Enterpriseプランは30分です）



（※Business, Enterpriseプランは30分です）

【初級編】
【中級編】
【上級編】

 【初級編】

【中級編】

【上級編】

【初級編】
スプレッドシートの
やり取りを高速化

都度取得と一括取得で比較

```
//①各セルをgetRange()で都度取得し、
//そのセルの値をgetValue()で都度取得する場合
function getSpeadsheet_each(){

 var spreadsheetId = '{your_spreadsheet_id}';
 var file = SpreadsheetApp.openById(spreadsheetId);
 var sheet = file.getSheetByName('{your_sheet_name}');


 var buf = '';
 for (var iRow = 1; iRow <= sheet.getLastRow(); iRow++){
  for (var iCol = 1; iCol <= sheet.getLastColumn(); iCol++){
   buf += sheet.getRange(iRow, iCol).getValue();
  }
 }
}
```

```
//②getDataRange().getValues()で一括取得する場合

function getSpeadsheet_bulk(){

 var spreadsheetId = '{your_spreadsheet_id}';
 var file = SpreadsheetApp.openById(spreadsheetId);
 var sheet = file.getSheetByName('{your_sheet_name}');
 var table = sheet.getDataRange().getValues();

 var buf = '';
 for (var iRow = 0; iRow < table.length; iRow++){
  for (var iCol = 0; iCol < table[0].length; iCol++){
   buf += table[iRow][iCol];
  }
 }
}
```

どちらか速く全セルの値を結合できるか

都度取得
getRange().getValue()

```
//①各セルをgetRange()で都度取得し、
//そのセルの値をgetValue()で都度取得する場合
function getSpeadsheet_each(){

 var spreadsheetId = '{your_spreadsheet_id}';
 var file = SpreadsheetApp.openById(spreadsheetId);
 var sheet = file.getSheetByName('{your_sheet_name}');


 var buf = '';
 for (var iRow = 1; iRow <= sheet.getLastRow(); iRow++){
  for (var iCol = 1; iCol <= sheet.getLastColumn(); iCol++){
   buf += sheet.getRange(iRow, iCol).getValue();
  }
 }
}
```

一括取得
getDataRange().getValues()

```
//②getDataRange().getValues()で一括取得する場合

function getSpeadsheet_bulk(){

 var spreadsheetId = '{your_spreadsheet_id}';
 var file = SpreadsheetApp.openById(spreadsheetId);
 var sheet = file.getSheetByName('{your_sheet_name}');
 var table = sheet.getDataRange().getValues();

 var buf = '';
 for (var iRow = 0; iRow < table.length; iRow++){
  for (var iCol = 0; iCol < table[0].length; iCol++){
   buf += table[iRow][iCol];
  }
 }
}
```

今回は、11行×1列のシートを対象に、
5回テストした平均値で比較しました。

都度取得
getRange().getValue()

```
//①各セルをgetRange()で都度取得し、
//そのセルの値をgetValue()で都度取得する場合
function getSpeadsheet_each(){

 var spreadsheetId = '{your_spreadsheet_id}';
 var file = SpreadsheetApp.openById(spreadsheetId);
 var sheet = file.getSheetByName('{your_sheet_name}');


 var buf = '';
 for (var iRow = 1; iRow <= sheet.getLastRow(); iRow++){
  for (var iCol = 1; iCol <= sheet.getLastColumn(); iCol++){
   buf += sheet.getRange(iRow, iCol).getValue();
  }
 }
}
```

4.094秒

一括取得
getDataRange().getValues()

```
//②getDataRange().getValues()で一括取得する場合

function getSpeadsheet_bulk(){

 var spreadsheetId = '{your_spreadsheet_id}';
 var file = SpreadsheetApp.openById(spreadsheetId);
 var sheet = file.getSheetByName('{your_sheet_name}');
 var table = sheet.getDataRange().getValues();

 var buf = '';
 for (var iRow = 0; iRow < table.length; iRow++){
  for (var iCol = 0; iCol < table[0].length; iCol++){
   buf += table[iRow][iCol];
  }
 }
}
```

0.476秒

圧勝

スプレッドシートやシート、セルの関数は
とにかく遅いのでなるべく使わない。

使うとしても、最小回数にする。

【初級編】

 【中級編】

【上級編】

【中級編】
スプレッドシートの
限界を超える

～悲しいスプレッドシートの限界～

1. 1セルには**半角5万文字**が上限。
2. 全シート合計で、**200万セル**が上限。
3. 入力先はセルなので、
   「=」から始まる文字列を入力しようとすると高確率で数式エラーになってGASが強制終了する。

～悲しい打開策～

1. 半角5万文字を超える文字列は、
   複数のセルに**分割保存**
2. 200万セルを超えそうなデータは、
   複数のスプレッドシートに**分割保存**
3. 予期せぬ数式エラーを防ぐため、
   冒頭に「#」など**適当な文字を挿入**

～悲しい打開策～

1. 半角5万文字を超える文字列は、
   本当にスプレッドシートで
   良いのだろうか？
2. 200万セルを超えるようなデータは、
   複数のスプレッドシートに分割保存
3. 予期せぬ数式エラーを防ぐため、
   冒頭に「＃」など適当な文字を挿入

【初級編】

【中級編】

 【上級編】

Google
ドキュメント

Google
ドライブ
（テキスト）

Google
Apps Script
（プロパティサービス）

Google
ドキュメント

```
//③Googleドキュメントからのデータ取得
function getDocument(){

  var id = '{your_document_id}';
  var file = DocumentApp.openById(id);
  var text = file.getBody().getText();

}
```

Google
ドライブ
（テキスト）

```
//④Googleドライブのテキストからのデータ取得
function getText(){

  var id = '{your_text_id}';
  var file = DriveApp.getFileById(id);
  var text = file.getBlob ().getDataAsString ();

}
```

Google Apps Script（プロパティサービス）

```
//⑥GASのプロパティサービスからのデータ取得
function getProperty(){

  var properties = PropertiesService.getScriptProperties();
  var text = properties['{your_property_name}'];

}
```

データ容量別の速度がこちら

	データ上限	読み込み速度			
		小データ (1KB)	基準容量 (716KB)	基準容量×2倍 (≒102万文字)	基準容量×71倍 (≒50MB)
Googleスプレッドシート	・半角5万文字／セル ・200万セル／ファイル	0.376 秒	0.476 秒	0.545 秒	13.539 秒
Googleドキュメント	・半角102万文字／ファイル	0.084 秒	0.203 秒	0.443 秒	-
Googleドライブ （テキスト保存）	・50MB／ファイル	0.425 秒	0.404 秒	0.481 秒	3.394 秒
プロパティサービス （GAS内部）	・9KB／プロパティ ・500KB／ファイル ・文字列として保存される。	0.001 秒	-	-	-

データ容量別の速度がこちら

	データ上限	読み込み速度			
		小データ (1KB)	基準容量 (716KB)	基準容量×2倍 (≒102万文字)	基準容量×71倍 (≒50MB)
Googleスプレッドシート	・半角5万文字／セル ・200万セル／ファイル	0.376 秒	0.476 秒	0.545 秒	13.539 秒
Googleドキュメント	・半角102万文字／ファイル	0.084 秒	0.203 秒	0.443 秒	-
Googleドライブ （テキスト保存）	・50MB／ファイル	0.425 秒	0.404 秒	0.481 秒	3.394 秒
プロパティサービス （GAS内部）	・9KB／プロパティ ・500KB／ファイル ・文字列として保存される。	0.001 秒	-	-	-

データ容量別の速度がこちら

	データ上限	読み込み速度			
		小データ (1KB)	基準容量 (716KB)	基準容量×2倍 (≒102万文字)	基準容量×71倍 (≒50MB)
Googleスプレッドシート	・半角5万文字／セル ・200万セル／ファイル	0.376 秒	0.476 秒	0.545 秒	13.539 秒
Googleドキュメント	・半角102万文字／ファイル	0.084 秒	0.203 秒	0.443 秒	-
Googleドライブ （テキスト保存）	・50MB／ファイル	0.425 秒	0.404 秒	0.481 秒	3.394 秒
プロパティサービス （GAS内部）	・9KB／プロパティ ・500KB／ファイル ・文字列として保存される。	0.001 秒	-	-	-

～速さ重視の使い分け～

1. **9KB未満**の短いパラメータは
   プロパティサービス

2. **50MB未満**なら
   Googleドライブ（テキスト）

3. **人が手で更新**するデータ
   （あまり大きくないものに限る）は、
   Googleスプレッドシート

所感的まとめ

結果としてはプロパティサービスの高速さだけでなく、
Googleドライブ（テキスト）の意外な可能性が見えた分析になりましたが、このテキスト形式はアプリ自体にエディタがなく、データの更新は原則としてGAS経由で行わなければならないというデメリットもあります。

色々と考えた先には、
(お金払ってGoogle Cloud Platform使うのが最強じゃん…)
などという元も子もない結論が垣間見えますが、それはそれ、これはこれということで、課金なしでできる範囲を検証したのが今回でございます。

あなたの何かの気付きにつながれば幸いです。

ご清聴
ありがとうございました