eBook Gratuit

APPRENEZ google-apps-script

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#google-apps-script

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: google-apps-script

It is an unofficial and free google-apps-script ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official google-apps-script.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec google-apps-script

Remarques

L'aperçu officiel de Google Apps Script est publié à l' adresse http://www.google.com/script/start , à partir de là.

> Google Apps Script est un langage de script JavaScript en nuage qui permet d'automatiser facilement les tâches sur les produits Google et les services tiers et de créer des applications Web.

De https://developers.google.com/apps-script/guides/services/#basic_javascript_features

> Apps Script est basé sur JavaScript 1.6 , plus quelques fonctionnalités de 1.7 et 1.8 . Outre les services Google intégrés et avancés, de nombreuses fonctionnalités JavaScript de base sont disponibles: vous pouvez utiliser des objets communs tels que Array , Date , RegExp , etc. , ainsi que les objets globaux Math et Object . Cependant, le code Apps Script s'exécutant sur les serveurs de Google (pas côté client, sauf pour les pages de service HTML), les fonctionnalités basées sur navigateur telles que la manipulation DOM ou l'API Windows ne sont pas disponibles.

Examples

Installation ou configuration

Le script Google Apps ne nécessite ni installation ni configuration. La seule exigence est un compte Google. Un compte Gmail fonctionne aussi bien qu'un compte Google Apps for Work / Éducation / Gouvernement. Vous pouvez créer un nouveau compte Google en accédant à accounts.google.com

Commencez votre premier script en allant sur script.google.com . Vous pouvez également accéder à Google Apps Script sous les `tools -> Script editor...` de nombreuses applications Google, c.-à-d. *Docs, Sheets, Forms, etc.* Google Apps Script peut également être ajouté directement à votre Google Drive avec la fonctionnalité `Connect more apps...` .

La documentation officielle est disponible sur developers.google.com/apps-script/ .

Pour que les scripts d'application s'exécutent, ils doivent contenir un fichier code.gs. Le fichier code.gs doit contenir une fonction nommée doGet (scripts autonomes) ou une fonction onOpen (scripts addon). Les démarrages rapides dans la documentation contiennent des exemples.

Si une API est activée dans le script d'application, elle doit également être activée dans la console du développeur. Cependant, la console des développeurs contient des API qui peuvent être activées mais n'apparaissent pas dans l'interface du script d'application. Par exemple, Marketplace SDK doit être activé dans la console des développeurs avant que l'application puisse être publiée sur Google Play Store ou sur un déploiement à l'échelle du domaine de la suite G.

Pour les applications Google pour l'éducation / le travail / le gouvernement, il existe des paramètres dans la console d'administration du domaine qui peuvent être ajustés pour autoriser ou interdire l'exécution de scripts d'application.

Types de scripts

Les scripts Google App sont de trois types.

- Autonome
- Lié à Google Apps
- Applications Web

Script autonome

Les scripts autonomes ne sont liés à aucune application Google, par exemple *Docs, Sheets ou Forms, etc. Vous* pouvez créer un script autonome en visitant *script.google.com* ou en connectant un script d'application Google à Google Drive. Le script autonome peut être utilisé pour programmer des applications Google indépendamment, peut être utilisé comme une application Web ou peut être configuré pour s'exécuter automatiquement à partir d'un déclencheur installable. Voir la documentation du script autonome.

Lié à Google Apps

Script lié à Google Apps, également appelé script lié à un conteneur; Contrairement aux scripts autonomes, ils sont liés aux applications Google, par exemple *Google Docs ou Google Sheets, etc. Vous* pouvez créer un script lié à un conteneur en sélectionnant des `tools> Script editor` de `tools> Script editor` dans Google App. Certaines fonctionnalités telles que les boîtes de dialogue, les invites, les menus et la barre latérale ne sont fournies que par les scripts liés aux conteneurs. En outre, un script lié à un conteneur est utilisé pour créer des modules complémentaires Google . Voir la documentation des scripts liés aux conteneurs.

Applications Web

Google App Script peut être utilisé en tant qu'application Web accessible par navigateur. Web App peut fournir une interface utilisateur sur le navigateur et utiliser des applications Google, par exemple des *documents, des feuilles, etc.* Les scripts et scripts autonomes liés à Google Apps peuvent être transformés en applications Web. Pour qu'un script fonctionne comme une application Web, le script doit répondre à deux exigences:

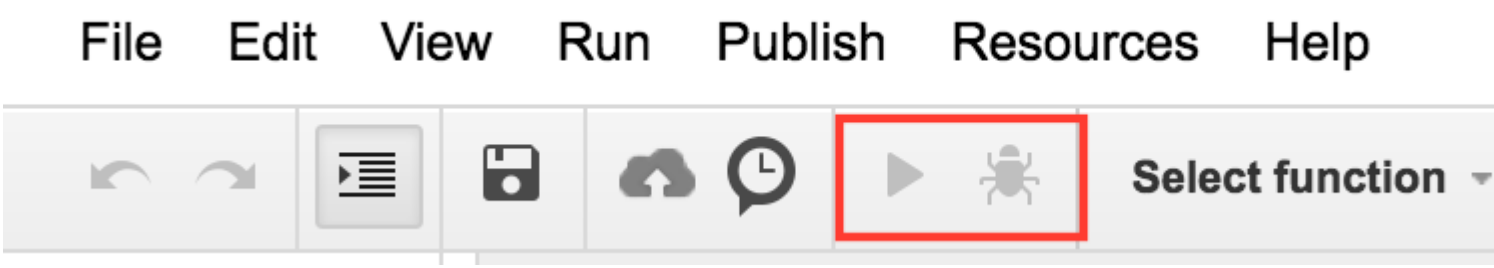
- inclure une fonction `doGet()` ou `doPost()` .
- La fonction renvoie un objet HTML service HtmlOutput ou un objet Content Service TextOutput.

Les fonctions Inshort, `doGet()` et `doPost()` fonctionnent respectivement comme les gestionnaires de requêtes et de requêtes HTTP.

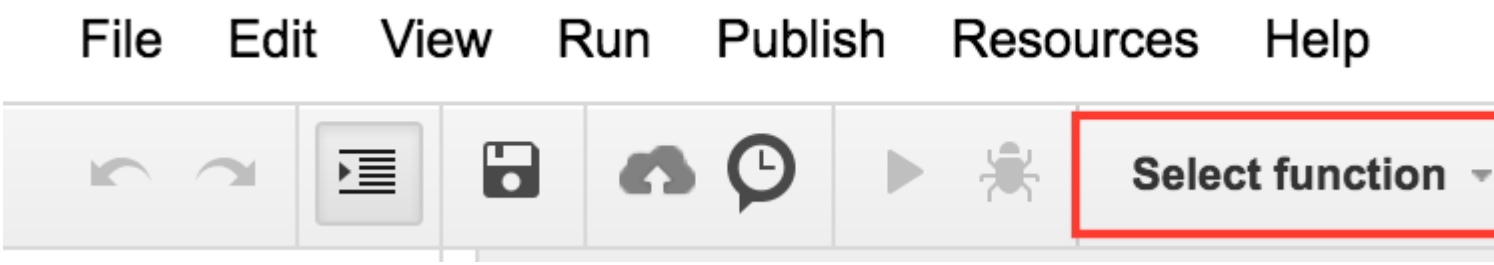
Pour plus de détails sur les applications Web, consultez la documentation officielle.

Lancer / déboguer votre script

Essayez d'exécuter votre code à partir de la barre d'outils, comme indiqué ci-dessous:



Dans votre code, si vous avez plus d'une fonction, vous devez indiquer la fonction que vous souhaitez utiliser avant de l'exécuter. Par exemple :



Vous pouvez également appuyer sur **ctrl + r** depuis votre clavier pour exécuter le code. Il enregistrera d'abord le code, s'il n'est pas enregistré, puis l'exécutera. Mais pour que cela fonctionne, vous devez avoir sélectionné la fonction, comme indiqué dans l'image ci-dessus.

De plus, si votre script est appelé par des activités externes, vous pourrez toujours voir les journaux en cliquant sur view-> logs si vous enregistrez quelque chose après l'exécution du code.

Bonjour le monde

Nous allons dire bonjour comme un message.

```
function helloWorld()
{
  Browser.msgBox("Hello World");
}
```

Pour exécuter le script, cliquez sur ▶ ou sélectionnez l'élément de menu **Exécuter** -> **helloWorld**

Un examen plus approfondi du script Google Apps

Google Apps Script est une plate-forme basée sur JavaScript, principalement utilisée pour automatiser et étendre Google Apps. Apps Script s'exécute exclusivement sur l'infrastructure de Google ne nécessitant aucun provisionnement ni configuration du serveur. Un IDE en ligne sert d'interface à l'ensemble de la plate-forme connectant tous les services disponibles à Apps Script. L'authentification de l'utilisateur est intégrée à la plate-forme via OAuth2 et ne nécessite aucun code ou configuration par l'auteur du script.

Apps Script s'exécute côté serveur, mais peut avoir des interfaces utilisateur créées avec HTML,

CSS, JavaScript ou toute autre technologie prise en charge par le navigateur. Contrairement aux Nodejs, qui sont pilotés par des événements, les scripts d'application s'exécutent dans un modèle threadé. Tous les appels à un script génèrent une instance unique de ce script qui s'exécute indépendamment de toutes les autres instances. Lorsqu'une instance d'un script termine son exécution, elle est détruite.

Les fonctions du script Apps bloquent les fonctions de callback et de programmation asynchrone. Le verrouillage est utilisé pour empêcher l'exécution simultanée de sections de code critiques, telles que le fichier IO, par différentes instances.

En pratique, l'écriture d'applications est simple. Vous trouverez ci-dessous un script simple qui crée une nouvelle feuille de calcul à partir d'une feuille de calcul de modèle.

```
// Create a new spreadsheet from a template
function createSpreadsheet(){
   var templateFileId = '1Azcz9GwCeHjGl9TXf4aUh6g20Eqmgd1UMSdNVjzIZPk';
   var sheetName = 'Account Log for:' + new Date();
   SpreadsheetApp.openById(templateFileId).copy(sheetName);
}
```

Lire Démarrer avec google-apps-script en ligne: https://riptutorial.com/fr/google-apps-script/topic/1154/demarrer-avec-google-apps-script

Chapitre 2: Appels clients à Google apps-script

Introduction

Google appscript fonctionne bien en tant que plate-forme autonome et au format addon pour les documents, feuilles et formulaires Google. Cependant, il peut arriver qu'un navigateur client doive appeler une application Google pour effectuer certaines actions.

Par conséquent, Google a introduit des demandes côté client dans des scripts d'applications Google. Pour résoudre ce problème, Google a introduit les bibliothèques côté client

Examples

Voici un exemple d'appel côté client vers un script d'application Google

```
<script src="https://apis.google.com/js/api.js"></script>
<script>
function start() {
 // 2. Initialize the JavaScript client library.
 gapi.client.init({
'apiKey': 'YOUR_API_KEY',
// clientId and scope are optional if auth is not required.
'clientId': 'YOUR_WEB_CLIENT_ID.apps.googleusercontent.com',
'scope': 'profile',
}).then(function() {
// 3. Initialize and make the API request.
return gapi.client.request({
  'path': 'https://people.googleapis.com/v1/people/me',
  })
 }).then(function(response) {
console.log(response.result);
}, function(reason) {
console.log('Error: ' + reason.result.error.message);
});
};
// 1. Load the JavaScript client library.
 gapi.load('client', start);
</script>
```

Lire Appels clients à Google apps-script en ligne: https://riptutorial.com/fr/google-apps-script/topic/8875/appels-clients-a-google-apps-script

Chapitre 3: Apps Script Web Apps

Remarques

Ceci est un exemple d'application Web de formulaire, le bit côté client montre une conception UX de base, telle qu'un bouton d'envoi désactivé lorsque le formulaire est soumis, ou un message d'erreur s'il échoue ... etc

Le bit Apps Script est très simple. Il ne contient que le code nécessaire pour servir le code HTML et pour valider le champ.

Voici un lien vers cet exemple d'application en action: Exemple de formulaire de script d'applications

Remarque: vous devez être connecté à un compte Google.

La structure du fichier de script d'applications est la suivante:

- Code.gs
- index.html
- Stylesheet.html
- JavaScript.html

Examples

Formulaire d'application Web

Script d'applications:

```
//Triggered when the page is navigated to, serves up HTML
function doGet(){
  var template = HtmlService.createTemplateFromFile('index');
  return template.evaluate()
      .setTitle('Example App')
      .setSandboxMode(HtmlService.SandboxMode.IFRAME);
}

//Called from the client with form data, basic validation for blank values
function formSubmit(formData){
  for(var field in formData){
    if(formData[field] == ''){
      return {success: false, message: field + ' Cannot be blank'}
    }
  }
  return {success: true, message: 'Sucessfully submitted!'};
}
```

HTML

```
<!DOCTYPE html>
<html>

    <head>
        <base target="_top">
        <link href="https://ssl.gstatic.com/docs/script/css/add-ons1.css" rel="stylesheet">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js"
type="text/javascript"></script>
    </head>

    <body>
        <div id="mainForm">
            <h1>Example Form</h1>
            <form>
                <div>
                    <div class="inline form-group">
                        <label for="name">Name</label>
                        <input id="nameInput" style="width: 150px;" type="text">
                    </div>
                </div>
                <div>
                    <div class="inline form-group">
                        <label for="city">City</label>
                        <input id="cityInput" style="width: 150px;" type="text">
                    </div>
                    <div class="inline form-group">
                        <label for="state">State</label>
                        <input id="stateInput" style="width: 40px;" type="text">
                    </div>
                    <div class="inline form-group">
                        <label for="zip-code">Zip code</label>
                        <input id="zip-codeInput" style="width: 65px;" type="number">
                    </div>
                </div>
                <div class="block form-group">
                    <label for="typeSelect">Type</label>
                    <select id="typeSelect">
                        <option value="">
                        </option>
                        <option value="Type 1 ">
                            Type 1
                        </option>
                        <option value="Type 2 ">
                            Type 2
                        </option>
                        <option value="Type 3 ">
                            Type 3
                        </option>
                        <option value="Type 4 ">
                            Type 4
                        </option>
                    </select>
                </div>
                <button class="action" id="submitButton" type="button">Submit</button>
                <button class="clear" id="clearFormButton" type="button">Clear Form</button>
            </form>
            <div class="hidden error message">
                <div class="title">Error:</div>
                <div class="message"></div>
            </div>
            <div class="hidden success message">
```

```
                <div class="title">Message:</div>
                <div class="message">Sucessfully submitted</div>
            </div>
        </div>
        <?!= HtmlService.createHtmlOutputFromFile('JavaScript').getContent(); ?>
        <?!= HtmlService.createHtmlOutputFromFile('Stylesheet').getContent(); ?>
    </body>

</html>
```

CSS

```
<style>
.hidden {
    display: none;
}

.form-group {
    margin: 2px 0px;
}

#submitButton {
    margin: 4px 0px;
}

body {
    margin-left: 50px;
}

.message {
   padding: 2px;
   width: 50%;
}

.message > * {
   display: inline-block;
}

.message .title {
    font-weight: 700;
    font-size: 1.1em;
}

.success.message {
    border: 1px solid #5c9a18;
    background: #e4ffe4;
    color: #2a8e2a;
}

.error.message {
    background: #f9cece;
    border: 1px solid #7d2929;
}

.error.message .title {
    color: #863030;
}

button.clear {
    background: -moz-linear-gradient(top, #dd6e39, #d17636);
```

```
    background: -ms-linear-gradient(top, #dd6e39, #d17636);
    background: -o-linear-gradient(top, #dd6e39, #d17636);
    background: -webkit-linear-gradient(top, #dd6e39, #d17636);
    background: linear-gradient(top, #dd6e39, #d17636);
    border: 1px solid transparent;
    color: #fff;
    text-shadow: 0 1px rgba(0, 0, 0, .1);
}

button.clear:hover {
    background: -moz-linear-gradient(top, #ca602e, #bd6527);
    background: -ms-linear-gradient(top, #ca602e, #bd6527);
    background: -o-linear-gradient(top, #ca602e, #bd6527);
    background: -webkit-linear-gradient(top, #ca602e, #bd6527);
    background: linear-gradient(top, #ca602e, #bd6527);
    border: 1px solid transparent;
    color: #fff;
    text-shadow: 0 1px rgba(0, 0, 0, .1);
}
</style>
```

JavaScript

```
<script>
var inputs = [
  'nameInput',
  'cityInput',
  'stateInput',
  'zip-codeInput',
  'typeSelect'
];

$(function(){
  var pageApp = new formApp();
  $('#submitButton').on('click', pageApp.submitForm);
  $('#clearFormButton').on('click', pageApp.clearForm);
});

var formApp = function(){
  var self = this;

  //Clears form input fields, removes message, enables submit
  self.clearForm = function(){
    for(var i = 0; i < inputs.length; i++){
        $('#'+inputs[i]).val('');
    }
    toggleSubmitButton(false);
    setErrorMessage(false);
    setSuccessMessage(false);
  }

  //Submits the form to apps script
  self.submitForm = function(){
    toggleSubmitButton(true);
    setSuccessMessage(false);
    setErrorMessage(false);

    google.script.run
        .withSuccessHandler(self.sucessfullySubmitted)
        .withFailureHandler(self.failedToSubmit)
```

```
        .formSubmit(self.getFormData());
  };

  //Retrieves the form data absed on the input fields
  self.getFormData = function(){
    var output = {};
    for(var i = 0; i < inputs.length; i++){
        output[inputs[i]] = $('#'+inputs[i]).val();
    }
    console.log(output)
    return output;
  }

  //When the apps script sucessfully returns
  self.sucessfullySubmitted = function(value){
    if(value.success){
      setSuccessMessage(true, value.message);
    } else {
      setErrorMessage(true, value.message);
      toggleSubmitButton(false);
    }
  }

  //When the apps script threw an error
  self.failedToSubmit = function(value){
    toggleSubmitButton(false);
    setErrorMessage(true, value.message);
  }
}

//Disables/enables the submit button
function toggleSubmitButton(disabled){
  $('#submitButton').prop('disabled', disabled);
}

//Sets the general message box's message and enables or disabled the error box
function setSuccessMessage(show, message){
  if(show){
    $('.success.message').removeClass('hidden');
    $('.success.message .message').text(message);
  } else {
    $('.success.message').addClass('hidden');
    $('.success.message .message').text('');
  }
}

//Sets the error message box's message and enables or disabled the error box
function setErrorMessage(show, message){
  if(show){
    $('.error.message').removeClass('hidden');
    $('.error.message .message').text(message);
  } else {
    $('.error.message').addClass('hidden');
    $('.error.message .message').text('');
  }
}

function getFormData(){
  var output = {};
  for(var i = 0; i < inputs.length; i++){
      output[inputs[i]] = $('#'+inputs[i]).val();
```

```
  }
  return output;
}
</script>
```

Lire Apps Script Web Apps en ligne: https://riptutorial.com/fr/google-apps-script/topic/4874/apps-script-web-apps

Chapitre 4: Créer une fonction personnalisée pour Google Sheets

Introduction

Une fonction personnalisée dans Google Documents est liée à un document spécifique (et ne peut donc être utilisé que dans ce document).

Il doit donc être créé avec l'édition du script de ce document (Tools -> Script Editor). Une fois sauvegardé, il peut ensuite être utilisé comme toute autre formule de tableur classique.

Examples

Constante standard de gravité

Cette fonction renvoie la constante de gravité standard dans les unités d'accélération spécifiées (1 pour cm / s², 2 pour ft / s², 3 pour m / s²)

```
/**
 * Returns the standard gravity constant in the specified acceleration units
 * Values taken from https://en.wikipedia.org/wiki/Standard_gravity on July 24, 2016.
 *
 * @param  {number}  input  1 for cm/s², 2 for ft/s², 3 for m/s²
 *
 * @customfunction
 */
function sg(units_key) {
  var value;
  switch(units_key) {
    case 1:
      value = 980.665;
      break;
    case 2:
      value = 32.1740;
      break;
    case 3:
      value = 9.80665;
      break;
    default:
      throw new Error('Must to specify 1, 2 or 3');
  }
  return value;
}
```

Pour utiliser la fonction, il doit être lié à une feuille de calcul en utilisant l'éditeur de script (Outils -> Editeur de script ...). Une fois la fonction ajoutée, elle peut être utilisée comme n'importe quelle autre fonction Google en appelant la fonction dans la formule d'une cellule.

Notez comment la fonction apparaît en saisie semi-automatique lors de la saisie dans une formule. Cela est dû au commentaire multi-ligne au-dessus de la déclaration de fonction qui est

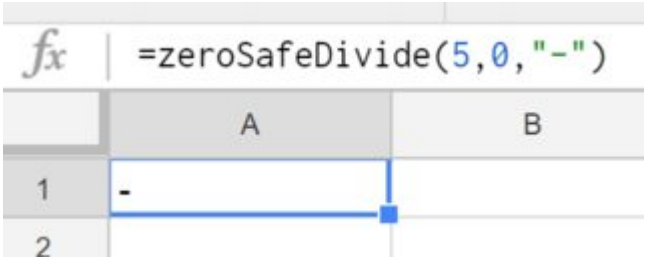
utilisé pour décrire ce que la fonction fait comme JSDoc et Javadoc. Pour que la formule s'affiche en autocomplétion, la balise @customfunction doit être spécifiée dans le commentaire.

Exemple de base

Pour éviter les erreurs inesthétiques `#DIV/0` dans une feuille de calcul, une fonction personnalisée peut être utilisée.

```
/**
 * Divides n by d unless d is zero, in which case, it returns
 * the given symbol.
 *
 * @param  {n}  number The numerator
 * @param  {d}  number The divisor
 * @param  {symbol}  string The symbol to display if `d == 0`
 * @return {number or string} The result of division or the given symbol
 *
 * @customfunction
 */
function zeroSafeDivide(n, d, symbol) {
  if (d == 0)
    return symbol;
  else
    return n / d;
}
```

Pour utiliser la fonction, il doit être lié à une feuille de calcul en utilisant l'éditeur de script (**Outils -> Editeur de script ...**). Une fois la fonction ajoutée, elle peut être utilisée comme n'importe quelle autre fonction Google en appelant la fonction dans la formule d'une cellule.



Notez comment la fonction apparaît en saisie semi-automatique lors de la saisie dans une formule. Cela est dû au commentaire multi-ligne au-dessus de la déclaration de fonction qui est utilisé pour décrire ce que la fonction fait comme JSDoc et Javadoc. Pour que la formule s'affiche en autocomplétion, la balise `@customfunction` doit être spécifiée dans le commentaire.

Lire Créer une fonction personnalisée pour Google Sheets en ligne: https://riptutorial.com/fr/google-apps-script/topic/5572/creer-une-fonction-personnalisee-pour-google-sheets

Chapitre 5: DriveApp

Examples

Créer un nouveau dossier dans une racine Google Drive

```
function createNewFolderInGoogleDrive(folderName) {
  return DriveApp.createFolder(folderName);
}
```

Utilisez la fonction `createNewFolderInGoogleDrive` pour créer un dossier nommé `Test folder` dans une racine Google Drive:

```
 var newFolder = createNewFolderInGoogleDrive('Test folder');
```

`newFolder` a le type Class Folder :

```
// output id of new folder to log
Logger.log(newFolder.getId());
```

Créer un nouveau fichier dans Google Drive d'un certain type Mime

```
 function createGoogleDriveFileOfMimeType() {
  var content,fileName,newFile;//Declare variable names

  fileName = "Test File " + new Date().toString().slice(0,15);//Create a new file name with
date on end
  content = "This is the file Content";

  newFile = DriveApp.createFile(fileName,content,MimeType.JAVASCRIPT);//Create a new file in
the root folder
};
```

Créer un nouveau fichier texte dans le dossier racine de Google Drive

```
 function createGoogleDriveTextFile() {
  var content,fileName,newFile;//Declare variable names

  fileName = "Test Doc " + new Date().toString().slice(0,15);//Create a new file name with
date on end
  content = "This is the file Content";

  newFile = DriveApp.createFile(fileName,content);//Create a new text file in the root folder
};
```

Créer un nouveau fichier dans Google Drive à partir d'un blob

```
 function createGoogleDriveFileWithBlob() {
```

```
  var blob,character,data,fileName,i,L,max,min,newFile,randomNmbr;//Declare variable names

  fileName = "Test Blob " + new Date().toString().slice(0,15);//Create a new file name with
date on end

  L = 500;//Define how many times to loop
  data = "";
  max = 126;
  min = 55;

  for (i=0;i<L;i+=1) {//Loop to create data
    randomNmbr = Math.floor(Math.random()*(max-min+1)+min);//Create a random number
    //Logger.log('randomNmbr: ' + randomNmbr);
    character = String.fromCharCode(randomNmbr);

    //Logger.log('character: ' + character);//Print the character to the Logs
    data = data + character;
  };

  blob = Utilities.newBlob(data, MimeType.PLAIN_TEXT, fileName);//Create a blob with random
characters

  newFile = DriveApp.createFile(blob);//Create a new file from a blob

  newFile.setName(fileName);//Set the file name of the new file
};
```

Récupère tous les dossiers - place les dossiers dans un jeton de continuation - puis récupère les jetons

```
 function processGoogleDriveFolders() {
  var arrayAllFolderNames,continuationToken,folders,foldersFromToken,thisFolder;//Declare
variable names

  arrayAllFolderNames = [];//Create an empty array and assign it to this variable name

  folders = DriveApp.getFolders();//Get all folders from Google Drive in this account
  continuationToken = folders.getContinuationToken();//Get the continuation token

  Utilities.sleep(18000);//Pause the code for 3 seconds

  foldersFromToken = DriveApp.continueFolderIterator(continuationToken);//Get the original
folders stored in the token
  folders = null;//Delete the folders that were stored in the original variable, to prove that
the continuation token is working

  while (foldersFromToken.hasNext()) {//If there is a next folder, then continue looping
    thisFolder = foldersFromToken.next();//Get the next folder
    arrayAllFolderNames.push(thisFolder.getName());//Get the name of the next folder
  };

  Logger.log(arrayAllFolderNames);//print the folder names to the Logs
};
```

Obtenez tous les fichiers - mettez-les dans un jeton de continuation - puis récupérez-les

```
 function processGoogleDriveFiles() {
  var arrayAllFileNames,continuationToken,files,filesFromToken,fileIterator,thisFile;//Declare
variable names

  arrayAllFileNames = [];//Create an empty array and assign it to this variable name

  files = DriveApp.getFiles();//Get all files from Google Drive in this account
  continuationToken = files.getContinuationToken();//Get the continuation token

  Utilities.sleep(18000);//Pause the code for 3 seconds

  filesFromToken = DriveApp.continueFileIterator(continuationToken);//Get the original files
stored in the token
  files = null;//Delete the files that were stored in the original variable, to prove that the
continuation token is working

  while (filesFromToken.hasNext()) {//If there is a next file, then continue looping
    thisFile = filesFromToken.next();//Get the next file
    arrayAllFileNames.push(thisFile.getName());//Get the name of the next file
  };

  Logger.log(arrayAllFileNames);
};
```

Ajouter un dossier au lecteur racine

```
function DriveAppAddFolder(child) {//Adds file to the root drive in Google Drive
  var body,returnedFolder;//Declare variable names

  if (!child) {
    body = "There is no folder";
    MailApp.sendEmail(Session.getEffectiveUser().getEmail(), "", "Error Adding Folder!", body)
    return;
  };

  returnedFolder = DriveApp.addFolder(child);//Add a folder to the root drive

  Logger.log('returnedFolder: ' + returnedFolder);//Print the folder results to the Logs
};


function createNewFolderInGoogleDrive() {
  var folder,newFolderName,timeStamp,dateTimeAsString;

  timeStamp = new Date();//Create a new date
  dateTimeAsString = timeStamp.toString().slice(0,15);

  newFolderName = 'Test Folder Name ' + dateTimeAsString;//Create new folder name with
date/time appended to name

  folder = DriveApp.createFolder(newFolderName);//Create a new folder
  DriveAppAddFolder(folder);//Call a function and pass a folder to the function
};
```

Créer un nouveau fichier texte et l'ajouter au dossier racine

```
 function DriveAppAddFile(child) {//Adds file to the root drive in Google Drive
```

```
  var body,returnedFolder;//Declare variable names

  if (!child) {
    body = "There is no file";
    MailApp.sendEmail(Session.getEffectiveUser().getEmail(), "", "Error Adding File!", body)
    return;
  };

  returnedFolder = DriveApp.addFile(child);

  Logger.log('returnedFolder: ' + returnedFolder);
};


function createNewFileInGoogleDrive() {
  var content,file,newFileName,timeStamp,dateTimeAsString;

  timeStamp = new Date();//Create a new date
  dateTimeAsString = timeStamp.toString().slice(0,15);

  content = "This is test file content, created at: " + dateTimeAsString;//Create content for
new file
  newFileName = 'Test File ' + dateTimeAsString;//Create new file name with date/time appended
to name

  file = DriveApp.createFile(newFileName, content);//Create a new file
  DriveAppAddFile(file);//Call a function and pass a file to the function
};
```

Obtenir tous les fichiers dans un dossier de lecteur

```
function onOpen() {

  // Add a custom menu to run the script
  var ss = SpreadsheetApp.getActiveSpreadsheet();
  var searchMenuEntries = [ {name: "Run", functionName: "search"}];
  ss.addMenu("Get Files", searchMenuEntries);
}

function getFiles() {

  // Get the active spreadsheet and the active sheet
  var ss = SpreadsheetApp.getActiveSpreadsheet();
  var ssid = ss.getId();

  // Look in the same folder the sheet exists in. For example, if this template is in
  // My Drive, it will return all of the files in My Drive.
  var ssparents = DriveApp.getFileById(ssid).getParents();
  var sheet = ss.getActiveSheet();

  // Set up the spreadsheet to display the results
  var headers = [["Last Updated", "File Owner", "File Name", "File URL"]];
  sheet.getRange("A1:D").clear();
  sheet.getRange("A1:D1").setValues(headers);

  // Loop through all the files and add the values to the spreadsheet.
  var folder = ssparents.next();
  var files = folder.getFiles();
  var i=1;
```

```
  while(files.hasNext()) {
    var file = files.next();
    if(ss.getId() == file.getId()){
      continue;
    }
    sheet.getRange(i+1, 1, 1,
4).setValues([[file.getLastUpdated(),file.getOwner().getName(),file.getName(),
file.getUrl()]]);
    i++;
  }
}
```

Lire DriveApp en ligne: https://riptutorial.com/fr/google-apps-script/topic/5363/driveapp

Chapitre 6: DriveApp - getFileById (id)

Remarques

Il est également possible d'obtenir un fichier par l'URL du fichier. L'ID d'un fichier se trouve dans l'URL. L'utilisation de l'ID au lieu de l'URL complète signifie que le paramètre est plus court. Stocker l'URL plutôt que l'ID prend plus de place.

Examples

Obtenir un fichier à partir de Google Drive à l'aide de l'ID de fichier

```
function getGoogleDriveFileById(id) {
  var file;

  file = DriveApp.getFileById(id);//Returns a file - The "id" must be a string

  //One way to manually get a file ID
  // - Open the file from Google Drive
  // - The file ID is in the URL in the browsers address bar
  //https://docs.google.com/spreadsheets/d/File_ID_is_here/edit#gid=0
};
```

Lire DriveApp - getFileById (id) en ligne: https://riptutorial.com/fr/google-apps-script/topic/6087/driveapp---getfilebyid--id-

Chapitre 7: DriveApp Service - Fichiers par type et chaîne de recherche

Paramètres

Le nom du paramètre	Utiliser pour
searchString	la chaîne à trouver dans le nom du fichier

Examples

Récupère les fichiers par type de fichier avec la chaîne correspondante dans le nom du fichier

Obtenez tous les formulaires Google avec le mot "Sans titre" dans le nom du fichier.

```
function mainSearchFunction(searchStr) {
  var fileInfo,arrayFileIDs,arrayFileNames,arrayOfIndexNumbers,
      allFileIDsWithStringInName,i,searchStr,thisID;//Declare variables

  if (!searchStr) {
    searchStr = "Untitled";//Assign a string value to the variable
  };

  fileInfo = getFilesOfType();//Run a function that returns files information
  arrayFileNames = fileInfo[1];//Get the array of file names
  arrayOfIndexNumbers = searchFileNamesForString(arrayFileNames,searchStr);

  //Logger.log('searchStr: ' + searchStr)
  //Logger.log(arrayOfIndexNumbers)

  allFileIDsWithStringInName = [];
  arrayFileIDs = fileInfo[0];

  for (i=0;i<arrayOfIndexNumbers.length;i+=1) {
    thisID = arrayFileIDs[arrayOfIndexNumbers[i]];
    allFileIDsWithStringInName.push(thisID);
  };

  Logger.log(allFileIDsWithStringInName)
};

function getFilesOfType() {
  var allFormFiles,arrFileName,arrFileID,arrFileUrls,thisFile;

  allFormFiles = DriveApp.getFilesByType(MimeType.GOOGLE_FORMS);
  arrFileName = [];
  arrFileID = [];
  arrFileUrls = [];

  while (allFormFiles.hasNext()) {
```

```
    thisFile=allFormFiles.next();
    arrFileName.push(thisFile.getName());
    arrFileID.push(thisFile.getId());
    arrFileUrls.push(thisFile.getUrl());
  };

  //Logger.log(arrFileName)
  return [arrFileID,arrFileName];
};


function searchFileNamesForString(arrayFileNames,searchStr) {
  var arrayIndexNumbers,i,L,thisName;

  arrayIndexNumbers = [];

  L = arrayFileNames.length;

  for (i=0;i<L;i+=1){
    thisName = arrayFileNames[i];
    Logger.log(thisName);
    Logger.log('thisName.indexOf(searchStr): ' + thisName.indexOf(searchStr));

    if (thisName.indexOf(searchStr) !== -1) {
      arrayIndexNumbers.push(i);
    };
  };

  return arrayIndexNumbers;
};
```

Lire DriveApp Service - Fichiers par type et chaîne de recherche en ligne: https://riptutorial.com/fr/google-apps-script/topic/4049/driveapp-service---fichiers-par-type-et-chaine-de-recherche

Chapitre 8: Feuille active SpreadsheetApp

Remarques

Méthode: getActive ()

Type de retour: Tableur

Examples

getActive () - Récupère une feuille de calcul active

Cela renvoie la feuille de calcul active ou null s'il n'y en a pas.

```
var currentSheet = SpreadsheetApp.getActive();
var url = currentSheet.getUrl();
Logger.log( url );
```

Lire Feuille active SpreadsheetApp en ligne: https://riptutorial.com/fr/google-apps-script/topic/5861/feuille-active-spreadsheetapp

Chapitre 9: Feuille de calcul Ajouter un menu

Syntaxe

1. addMenu (nom, sous-menu)

Paramètres

prénom	La description
prénom	le nom du menu à créer
sous-menus	un tableau de cartes JavaScript

Remarques

En règle générale, vous souhaiterez appeler addMenu à partir de la fonction onOpen afin que le menu soit automatiquement créé lors du chargement de la feuille de calcul.

```
 // The onOpen function is executed automatically every time a Spreadsheet is loaded
 function onOpen() {
   var activeSheet = SpreadsheetApp.getActiveSpreadsheet();
   var menuItems = [];
   // When the user clicks on "addMenuExample" then "Menu 1", the function Myfunction1 is
executed.
   menuItems.push({name: "Menu 1", functionName: "Myfunction1"});
   menuItems.push(null); // adding line separator
   menuItems.push({name: "Menu 2", functionName: "Myfunction2"});

   activeSheet.addMenu("addMenuExample", menuEntries);
 }
```

Examples

Créer un nouveau menu

Crée un nouveau menu dans l'interface utilisateur de la feuille de calcul. Chaque entrée de menu exécute une fonction définie par l'utilisateur.

```
   var activeSheet = SpreadsheetApp.getActiveSpreadsheet();
   var menuItems = [];
   // When the user clicks on "addMenuExample" then "Menu 1", the function Myfunction1 is
executed.
   menuItems.push({name: "Menu 1", functionName: "Myfunction1"});
   menuItems.push(null); // adding line separator
   menuItems.push({name: "Menu 2", functionName: "Myfunction2"});
```

```
   activeSheet.addMenu("addMenuExample", menuEntries);
```

Créer un menu personnalisé

/ *

Méthode: Créer un menu personnalisé C'est la première fonction à appeler lorsque l'application charge

---

* /

```
function onOpen() {
  var ui = SpreadsheetApp.getUi();
  // Or DocumentApp or FormApp.
  ui.createMenu('My HR')
      .addItem('Send Form to All', 'sendIDPForm_All')
      .addItem('Trigger IDP System', 'applyCategory')
      .addToUi();
}
```

Lire Feuille de calcul Ajouter un menu en ligne: https://riptutorial.com/fr/google-apps-script/topic/4253/feuille-de-calcul-ajouter-un-menu

Chapitre 10: Firebase et AppScript: Introduction

Introduction

Intégrez Firebase avec Google AppScript pour lire et écrire des données dans la base de données Firebase.

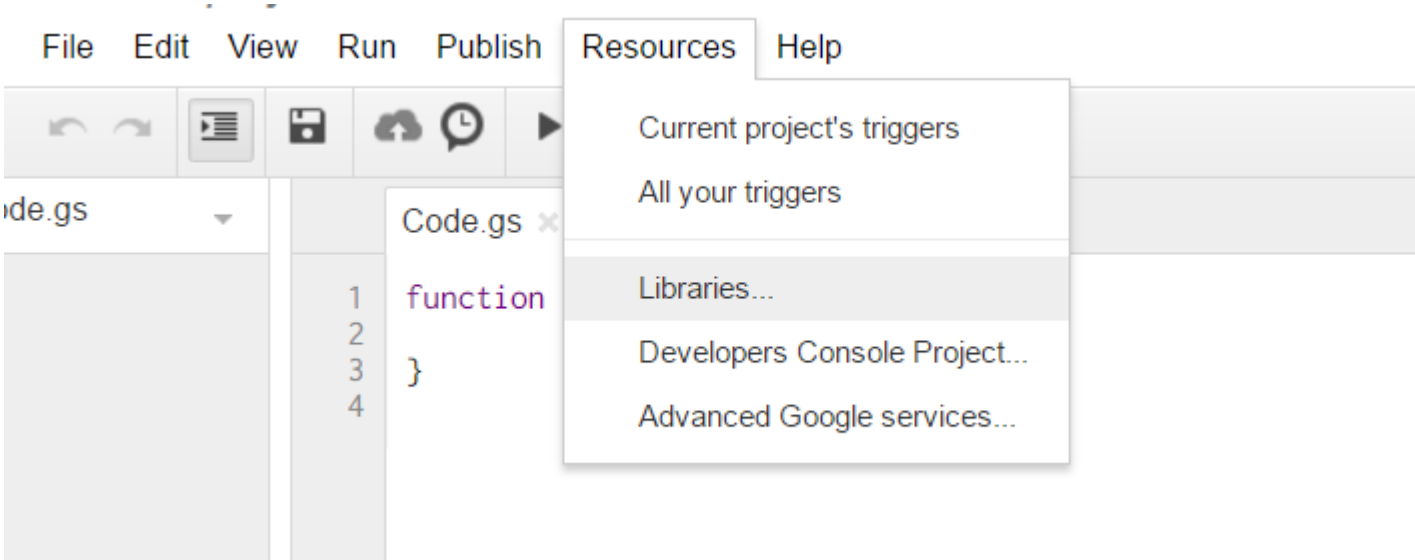
Firebase est un système de base de données NoSQL de Google qui utilise une base de données en temps réel pour créer et héberger des applications sur des appareils mobiles, de bureau et de tablette. Les bases de données NoSQL utilisent les objets JSON pour stocker les données au format structuré.

Examples

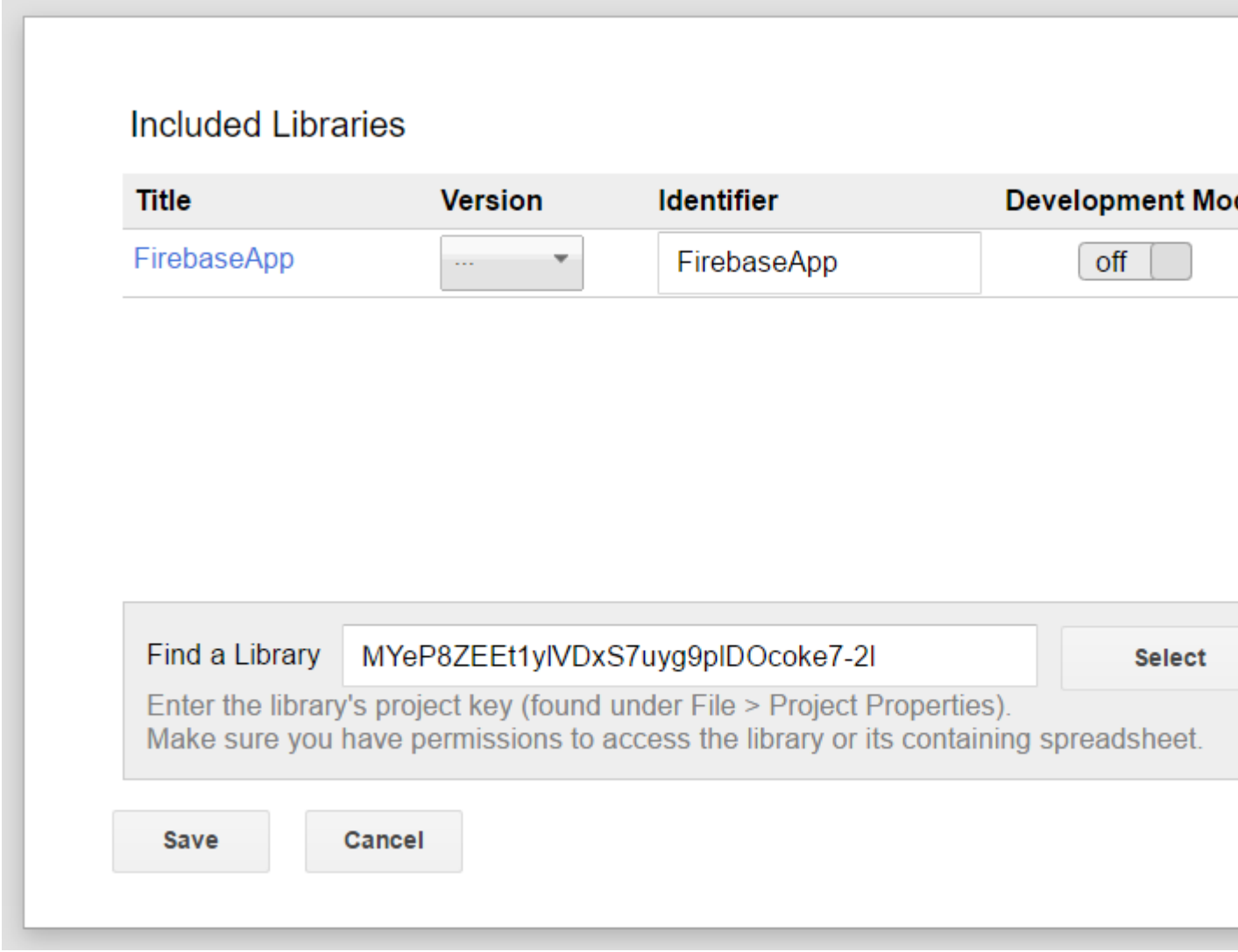
Connexion à un projet Firebase dans GAS et transfert de données de Google Spreadsheet vers Firebase

Installer la ressource Firebase dans AppScript

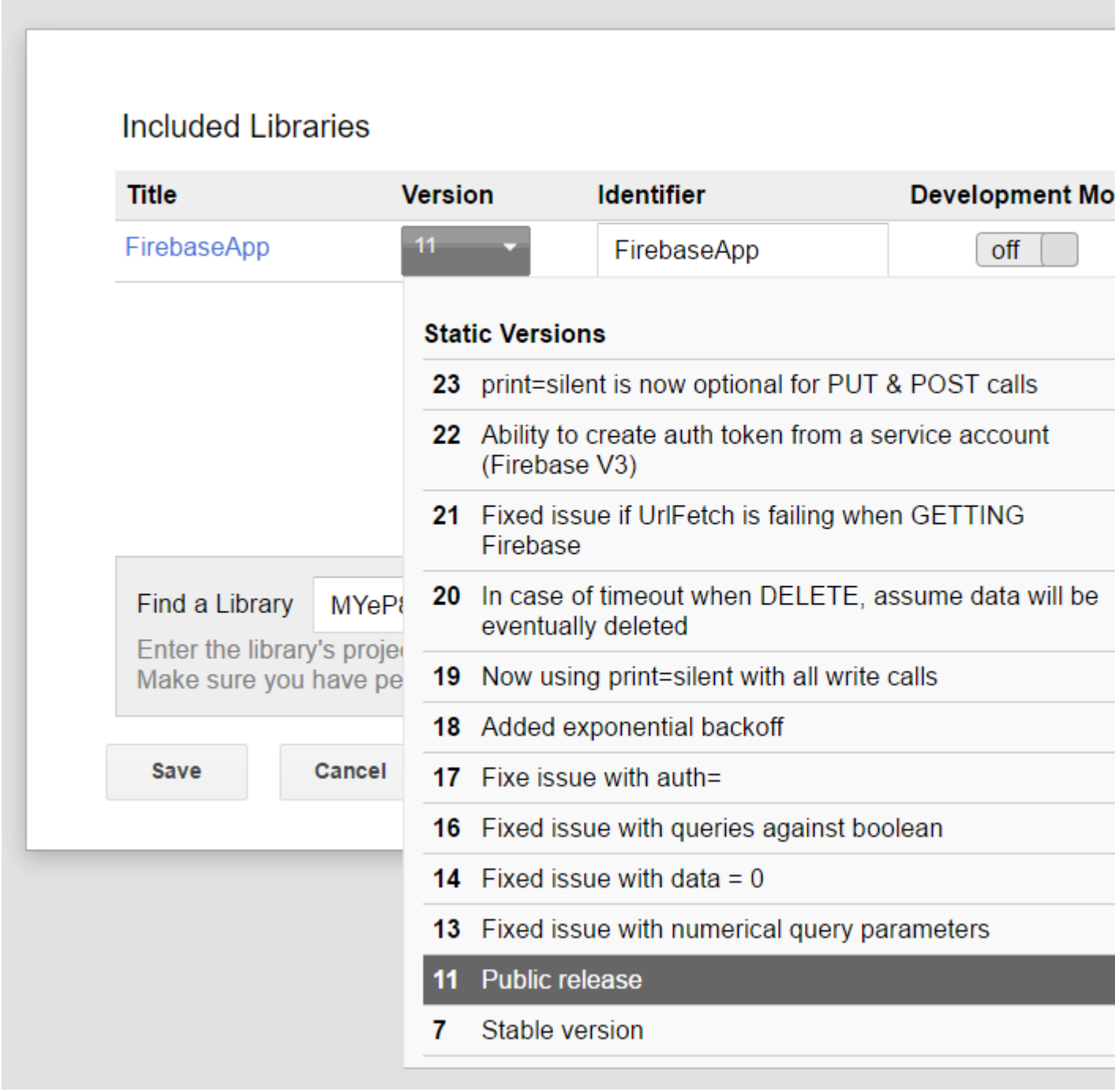
- Pour cela, cliquez sur Ressources puis sur Bibliothèques.
- Firebase possède une clé de bibliothèque de projet unique qui doit être installée dans AppScript.



- Cliquez sur Bibliothèques La fenêtre suivante apparaît. Entrez la clé de projet suivante dans la zone de texte. **MYeP8ZEEt1ylVDxS7uyg9plDOcoke7-2l** Ceci est la clé de bibliothèque de projet pour Firebase.



- Maintenant, dans la version, choisissez la version publique stable.



- Cliquez sur Enregistrer. Maintenant, Firebase est installé avec succès dans votre AppScript pour que vous puissiez travailler.

Prenons maintenant un exemple pour lire et écrire des données à partir de Firebase.

- Nous prenons maintenant un exemple de tableau conçu dans Google Sheets.

	A	B	C	D	E	
	First Name	Last Name	Email Address	Phone Number	Semester	Departm
	Vishal	vishwakarma	vishal.vishwakar	9594852468	7	INFT
	Yash	Udasi		75395185246	7	INFT

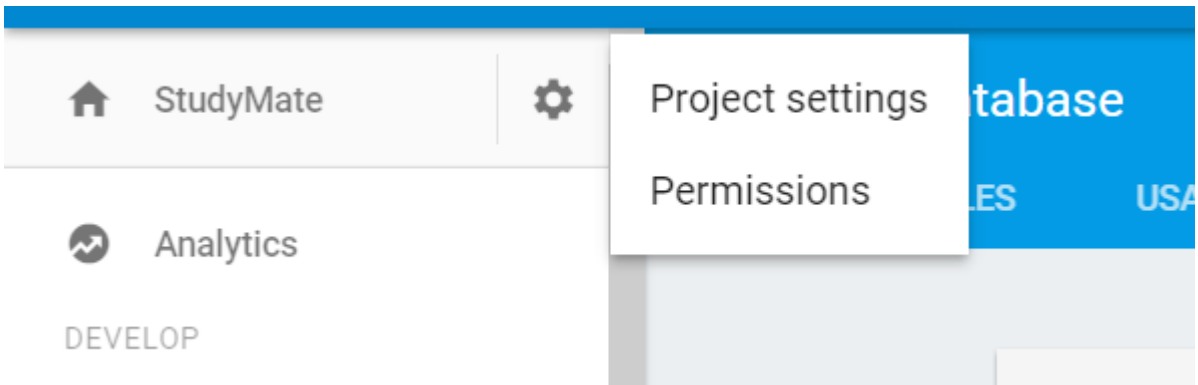
- Maintenant, construisez la base de données dans Firebase en utilisant cette table dans les feuilles. Ajoutez le code suivant dans AppScript.

```
function writeDataToFirebase() {
  var ss = SpreadsheetApp.openById("1LACsj0s3syAa9gvORdRWBhJ_YcXHybjQfHPgw3TLQ6g");
  var sheet = ss.getSheets()[0];
  var data = sheet.getDataRange().getValues();
  var dataToImport = {};
  for(var i = 1; i < data.length; i++) {
    var firstName = data[i][0];
    var lastName = data[i][1];
    dataToImport[firstName + '-' + lastName] = {
      firstName:firstName,
      lastName:lastName,
      emailAddress:data[i][2],
      semester:data[i][4],
      department:data[i][5],
    };
  }
  var firebaseUrl = "https://example-app.firebaseio.com/";
  var secret = "secret-key";
  var base = FirebaseApp.getDatabaseByUrl(firebaseUrl, secret);
  base.setData("", dataToImport);
}
```

Remplacez l'ID de feuille de calcul et le firebaseURL et la clé secrète.

Comment trouver le firebaseURL et la clé secrète?

- Accédez à votre tableau de bord Firebase et cliquez sur Paramètres en haut à gauche. Cliquez sur Paramètres du projet.



- Accédez à la section Comptes de service, vous pouvez trouver l'URL de base de données. Cela sert de firebaseURL.
- Cliquez maintenant sur l'onglet Secrets de base de données et vous pouvez trouver la clé secrète.

Vous avez maintenant inséré le firebaseURL et la clé secrète. Maintenant, vous êtes tous prêts à partir. Cliquez sur exécuter le code dans le moteur AppScript.

- Il vous demandera de revoir les autorisations lors de votre première exécution.
- Cliquez sur Autorisations de révision et Autoriser.
- Maintenant, vous exécutez votre fonction et vous pouvez voir la table créée dans la base de données Firebase.

Pour voir la base de données, accédez au tableau de bord Firebase et cliquez sur la base de données pour afficher la base de données.

Quelques fonctions supplémentaires à implémenter en lecture et en écriture.

1. Ecrire des données simples pour tester si la connexion fonctionne ou non.

```
function myFunction(){
 var firebaseUrl = "https://example-app.firebaseio.com/";
 var secret = "secret-key";
 var base = FirebaseApp.getDatabaseByUrl(firebaseUrl, secret);
 base.setData("test", "Hello Firebase");
}
```

2. Lire toutes les données

```
function getAllData() {
         var firebaseUrl = "https://example-app.firebaseio.com/";
         var secret = "secret-key";
var base = FirebaseApp.getDatabaseByUrl(firebaseUrl, secret);
 var data = base.getData();
          for(var i in data) {
            Logger.log(data[i].firstName + ' ' + data[i].lastName);
          }
}
```

Les données lues sont affichées dans les journaux. Pour vérifier les journaux, cliquez sur Cliquez sur Afficher → Journaux ou utilisez simplement Contrôle + Entrée.

3. Pour lire un enregistrement spécifique

```
function getContact() {
  var firebaseUrl = "https://example-app.firebaseio.com/";
  var secret = "secret-key";
  var base = FirebaseApp.getDatabaseByUrl(firebaseUrl, secret);
  var contact = base.getData("Yash-Udasi");
  Logger.log(contact);
}
```

Les données lues sont affichées dans les journaux. Pour vérifier les journaux, cliquez sur Cliquez sur Afficher → Journaux ou utilisez simplement Contrôle + Entrée.

4. Pour mettre à jour un enregistrement spécifique.

```
function updateData() {
  var firebaseUrl = "https://example-app.firebaseio.com/";
  var secret = "secret-key";
  var base = FirebaseApp.getDatabaseByUrl(firebaseUrl, secret);
  base.updateData("Yash-Udasi/emailAddress", "yash.udasi@fyuff.com");
}
```

Lire Firebase et AppScript: Introduction en ligne: https://riptutorial.com/fr/google-apps-script/topic/9417/firebase-et-appscript--introduction

Chapitre 11: GmailApp

Remarques

Voir aussi la référence API officielle pour GmailApp pour plus de détails sur les méthodes disponibles.

Examples

Obtenir le fichier CSV attaché à un mail

Supposons que nous ayons un système qui envoie des rapports quotidiens par courrier électronique sous la forme de fichiers CSV joints et que nous souhaitons y accéder.

```
function getCsvFromGmail() {
  // Get the newest Gmail thread based on sender and subject
  var gmailThread = GmailApp.search("from:noreply@example.com subject:\"My daily report\"", 0,
1)[0];

  // Get the attachments of the latest mail in the thread.
  var attachments = gmailThread.getMessages()[gmailThread.getMessageCount() -
1].getAttachments();

  // Get and and parse the CSV from the first attachment
  var csv = Utilities.parseCsv(attachments[0].getDataAsString());
  return csv;
}
```

Lire GmailApp en ligne: https://riptutorial.com/fr/google-apps-script/topic/5899/gmailapp

Chapitre 12: Google feuilles MailApp

Introduction

Ce service permet aux utilisateurs d'envoyer des e-mails avec un contrôle total sur le contenu de l'e-mail. Contrairement à GmailApp, le seul but de MailApp est d'envoyer des courriers électroniques. MailApp ne peut pas accéder à la boîte de réception Gmail d'un utilisateur.

Les modifications apportées aux scripts écrits à l'aide de GmailApp sont plus susceptibles de déclencher une demande de réautorisation d'un utilisateur que les scripts MailApp.

Examples

Un exemple de MailApp de base

MailApp est l'api de Google App Script qui peut être utilisé pour envoyer du courrier

```
function sendEmails() {

  var subject = "A subject for your new app!";
  var message = "And this is the very first message"
  var recipientEmail = "abc@example.com";

  MailApp.sendEmail(recipientEmail, subject, message);
}
```

La classe MailApp est limitée aux quotas basés sur votre compte Google:

- Utilisateur consommateur (compte personnel Gmail): 100 destinataires / jour
- Client Google Apps (ancien client): 100 destinataires / jour
- GSuite (basic / Gov / Edu / Business): 1500 destinataires / jour

Vous pouvez vérifier votre quota d'email dans `MailApp`

```
function checkQuota() {
  Logger.log(MailApp.getRemainingDailyQuota());
}
```

Accéder aux données de la feuille

```
function getSheetData() {

  var sheet = SpreadsheetApp.getActiveSheet();

  var startRow = 2;  // First row of data to process
  var numRows = 100;   // Number of rows to process
  var startCol = 1;  //First column of data to process
  var numCols = 15;    // Number of columns to process
```

```
  var dataRange = sheet.getRange(startRow, startCol, numRows, numCols);

  // Fetch values for each row in the Range.
  var data = dataRange.getValues();

  return data;
}
```

Vous pouvez également modifier la fonction ci-dessus pour obtenir une dynamique de plage de données à partir du contenu présent dans la feuille:

```
function getDataSheet() {

  sheet = SpreadsheetApp.getActiveSheet();

   //Get data range based on content
  var dataRange = sheet.getDataRange();

  // Fetch values for each row in the Range.
  var data = dataRange.getValues();

  return data;
}
```

Utiliser les données de feuille pour envoyer un courrier électronique

Compte tenu - Une feuille de renseignements sur les employés qui ont demandé un remboursement.

Besoin - Nous devrions envoyer un courriel à l'employé lorsque son remboursement est traité

Donc, la feuille est la suivante:

A	B	C
Name	Email Address	Reimberseme amount
Ramesh	ramesh@sample.com	20
Vidhita	vidhita@sample.com	5
Jhanvi	jhanvi@sample.com	3

La fonction d'envoi d'un email est la suivante:

```
function getDataSheet() {

  sheet = SpreadsheetApp.getActiveSheet();
```

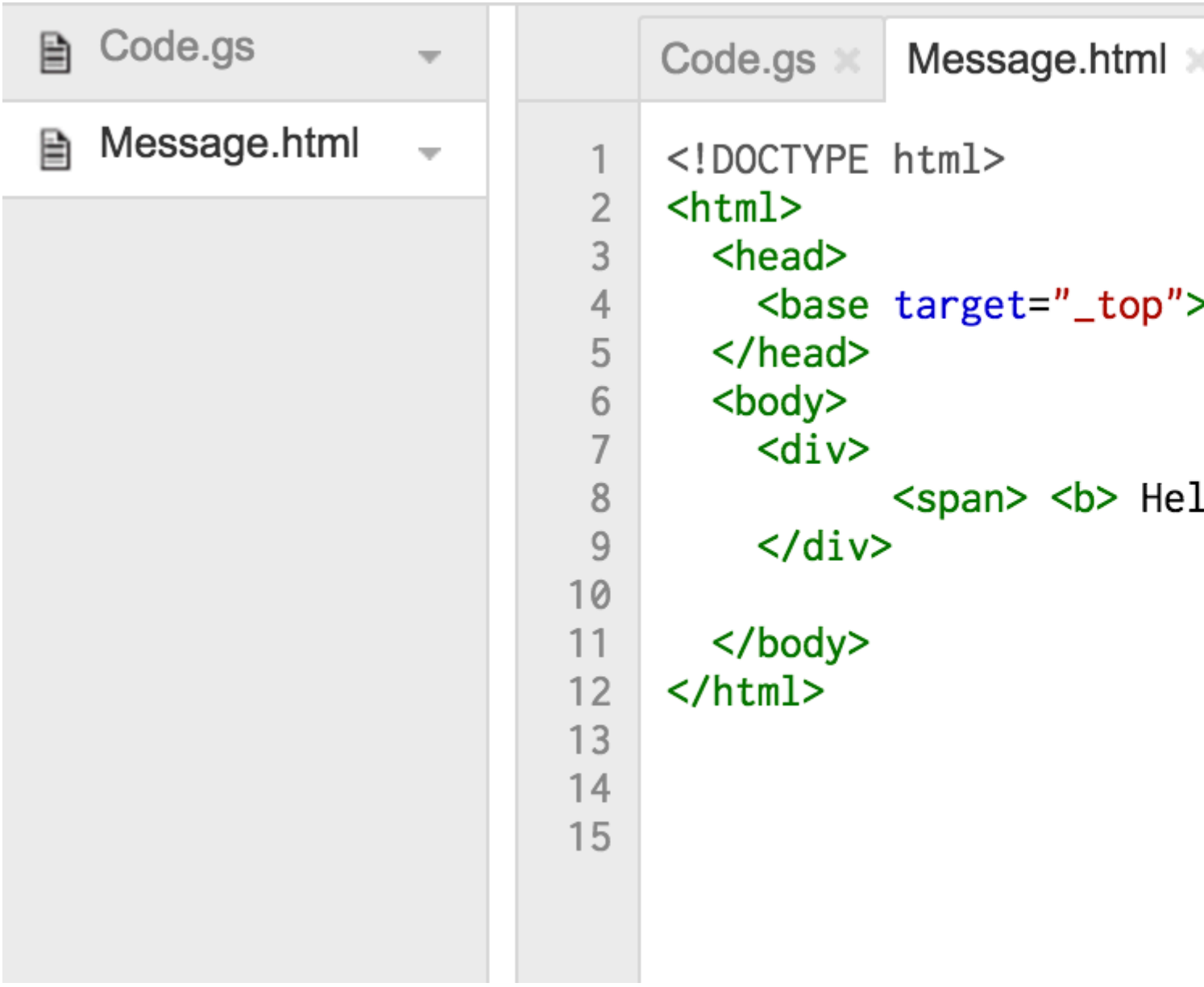
```
  startRow = 2;  // First row of data to process
  numRows = 100;   // Number of rows to process
  startCol = 1;  //First column of data to process
  numCols = 15;    // Number of columns to process

  var dataRange = sheet.getRange(startRow, startCol, numRows, numCols);

  // Fetch values for each row in the Range.
  var data = dataRange.getValues();

  return data;
}

function getMessage(name, amount) {
  return "Hello " + name + ", Your reimbursement for amount " + amount + " is processed
successfully";
}

function sendEmail() {

  var emailSent = "Yes";
  var reimbursed = "Yes";
  var emailCol = 5;

  var data = getDataSheet();

  for (var i = 0; i < data.length; i++) {

    var row = data[i];

    var isReimbursed = row[3];
    var isEmailSent = row[4];
    var name = row[0];
    var amount = row[2];

    if(isReimbursed == reimbursed && isEmailSent != emailSent) {

      var subject = "Reimbursement details";
      var message = getMessage(name, amount);

      var recipientEmail = row[1];

      MailApp.sendEmail(recipientEmail, subject, message);

      //Sheet range starts from index 1 and data range starts from index 0
      sheet.getRange(1 + i, emailCol).setValue(emailSent);
    }
  }
}
```

A	B	C
Name	Email Address	Reimberseme amount
Ramesh	ramesh@sample.com	20
Vidhita	vidhita@sample.com	5
Jhanvi	jhanvi@sample.com	3

Envoi de contenu HTML par courrier

Dans l'exemple ci-dessus, si nous voulons envoyer du contenu HTML en tant que message dans l'e-mail, créez un fichier HTML en allant dans **Fichier -> Nouveau -> Fichier HTML**

Maintenant, vous pouvez voir un fichier HTML en plus de votre fichier gs comme suit:



Maintenant, mettez à jour la méthode *getMessage ()* de l'exemple ci-dessus comme suit:

```
function getMessage(name, amount) {
  var htmlOutput = HtmlService.createHtmlOutputFromFile('Message'); // Message is the name of
the HTML file

  var message = htmlOutput.getContent()
  message = message.replace("%name", name);
  message = message.replace("%amount", amount);

  return message;
}
```

L'appel à l' *API MailApp* doit également être modifié

```
MailApp.sendEmail(recipientEmail, subject, message, {htmlBody : message});
```

Le code entier sera donc le suivant:

```
function getDataSheet() {

  sheet = SpreadsheetApp.getActiveSheet();

  startRow = 2;  // First row of data to process
  numRows = 100;   // Number of rows to process
  startCol = 1;  //First column of data to process
  numCols = 15;    // Number of columns to process

  var dataRange = sheet.getRange(startRow, startCol, numRows, numCols);

  // Fetch values for each row in the Range.
  var data = dataRange.getValues();

  return data;
}

function getMessage(name, amount) {
  var htmlOutput = HtmlService.createHtmlOutputFromFile('Message');

  var message = htmlOutput.getContent()
  message = message.replace("%name", name);
  message = message.replace("%amount", amount);

  return message;
}

function sendEmail() {

  var emailSent = "Yes";
  var reimbursed = "Yes";
  var emailCol = 5;

  var data = getDataSheet();

  for (var i = 0; i < data.length; i++) {

    var row = data[i];

    var isReimbursed = row[3];
    var isEmailSent = row[4];
    var name = row[0];
    var amount = row[2];

    if(isReimbursed == reimbursed && isEmailSent != emailSent) {

      var subject = "Reimbursement details";
      var message = getMessage(name, amount);

      var recipientEmail = row[1];

      MailApp.sendEmail(recipientEmail, subject, message, {htmlBody : message});

      sheet.getRange(startRow + i, emailCol).setValue(emailSent);
    }
  }
}
```

Lire Google feuilles MailApp en ligne: https://riptutorial.com/fr/google-apps-script/topic/5298/google-feuilles-mailapp

Chapitre 13: Script Google App Web à télécharger automatiquement à partir de Google Drive

Introduction

Ce script Web simple de Google App permet à Google Drive d'être interrogé à plusieurs reprises pour que les fichiers soient téléchargés sur le PC local de l'utilisateur. Montre comment utiliser un script d'application pour fournir les fonctions suivantes: 1. Interface utilisateur (simple dans cet exemple) 2. La page de téléchargement de fichier. Pour une explication plus complète de son fonctionnement, lisez l'exemple "Comment ça marche".

Remarques

Le script Web doit être publié pour pouvoir fonctionner.

Les fenêtres contextuelles doivent être activées pour https://script.google.com

Examples

forms.html

```
<!DOCTYPE html>
<html>
  <head>
    <base target="_top">
    <script>

    setInterval(
    function ()
    {
      document.getElementById('messages').innerHTML = 'Event Timer Fetching';
      google.script.run
        .withSuccessHandler(onSuccess)
        .withFailureHandler(onFailure)
        .fetchFromGoogleDrive();
    }, 60000);

    function callFetchGoogleDrive() {
      document.getElementById('messages').innerHTML = 'Fetching';
      google.script.run
        .withSuccessHandler(onSuccess)
        .withFailureHandler(onFailure)
        .fetchFromGoogleDrive();
    }

    function onSuccess(sHref)
    {
```

```
      if(new String(sHref).valueOf() == new String("").valueOf())
      {
        document.getElementById('messages').innerHTML = 'Nothing to download';
      }
      else
      {
        document.getElementById('messages').innerHTML = 'Success';
        document.getElementById('HiddenClick').href = sHref;
        document.getElementById('HiddenClick').click(); // Must enable Pop Ups for
https://script.google.com
      }
    }

    function onFailure(error)
    {
      document.getElementById('messages').innerHTML = error.message;
    }

    </script>
  </head>
  <body>
    <div id="messages">Waiting to DownLoad!</div>
    <div>
      <a id="HiddenClick" href="" target="_blank" onclick="google.script.host.close"
style="visibility: hidden;">Hidden Click</a>
    </div>
    <div>
      <button type="button" onclick='callFetchGoogleDrive();' id="Fetch">Fetch Now!</button>
    </div>
  </body>
</html>
```

code.gs

```
function doGet(e){
   var serveFile = e.parameter.servefile;
   var id = e.parameter.id;

   if(serveFile)
   {
     return downloadFile(id); // and Hyde
   }

  return HtmlService.createHtmlOutputFromFile('form.html'); // Jekyll
}

function fetchFromGoogleDrive() { // Jekyll
  var fileslist = DriveApp.searchFiles("your search criteria goes here + and trashed =
false"); // the 'and trashed = false' prevents the same file being download more than once

  if (fileslist.hasNext()) {
    var afile = fileslist.next();
    var html = ScriptApp.getService().getUrl()+"?servefile=true&id="+afile.getId();
    return html;
  }
  else
  {
    return '';
  }
```

```
}

function downloadFile(id){ // and Hyde
  try
  {
    var afile = DriveApp.getFileById(id);

    var aname = afile.getName();
    var acontent = afile.getAs('text/plain').getDataAsString();

    var output = ContentService.createTextOutput();
    output.setMimeType(ContentService.MimeType.CSV);
    output.setContent(acontent);
    output.downloadAsFile(aname);
    afile.setTrashed(true);
    return output;
  }
  catch (e) {
    return ContentService.createTextOutput('Nothing To Download')
  }
}
```

Comment ça marche

Application Web de Google Drive (autonome) pour télécharger automatiquement (interroger) les fichiers du lecteur vers le PC local de l'utilisateur (dossier Télécharger).

DriveApp fournit des mécanismes pour rechercher et télécharger des fichiers. Cependant, le mécanisme de téléchargement présente de sérieuses limitations en raison de l'architecture client / serveur héritée par Google Apps. (Pas de faute de Google)

DriveApp côté serveur ne fournit pas de fonction directe à télécharger sur le PC local car le serveur n’a aucune idée de l’endroit où se trouve le client et le téléchargement du fichier sur le serveur lui-même n’aurait aucune signification.

Le code côté serveur a besoin d'un mécanisme pour fournir au code côté client les données du fichier ou un lien vers le fichier. Ces deux mécanismes sont fournis mais les données du premier sont limitées à être utilisées directement par le code côté client. Le client n'a aucun mécanisme pour enregistrer les données, une fois obtenues, sur le PC local. Ainsi, il peut être utilisé pour afficher les données sur la page Web elle-même.

Le second mécanisme permet de renvoyer l'URL du script (lui-même) ou l'URL du fichier Drive. L'URL du fichier Drive n'est pas très utile car elle ne peut pas être utilisée directement dans le navigateur client pour télécharger le fichier. Placer cette URL dans l'ancre (et en cliquant dessus) ne donne lieu qu'à une page Web qui s'ouvre mais qui ne fait rien (sauf éventuellement afficher le fichier en ligne).

Cela laisse l'URL du script. Cependant, l'URL du script ne fournit que le script et non le fichier.

Pour lancer un téléchargement, le fichier du service Drive doit être renvoyé par la fonction doGet / doPost du script côté serveur à l'aide de ContentService createTextOutput, exactement comme indiqué dans les guides en ligne de Google. Cependant, cela implique qu'il ne peut y avoir aucun

autre élément d'interface utilisateur sur la page Web généré par les résultats renvoyés par doGet / doPost.

Cela nous laisse une solution très peu attrayante. Une page Web vierge sans éléments d'interface utilisateur qui télécharge une page, se ferme et nécessite une ouverture manuelle lorsqu'un autre téléchargement est requis.

De toute évidence, une autre page Web d'hébergement pourrait fournir l'interface utilisateur et le lien vers le script de téléchargement de l'application Web pour résoudre ce problème.

Ce script utilise une approche de Dr Jekyll et Mr Hyde pour résoudre ce problème.

Si le script est ouvert sans paramètres à GET (doGet), il affiche par défaut un formulaire. Ce sera la condition lorsque l'application publiée est ouverte pour la première fois par un utilisateur. Le formulaire fourni dans cet exemple est extrêmement simple.

Si le script est ouvert avec le paramètre servefile = true, le script se comporte comme un téléchargement de fichier Drive.

Le javascript côté client contient un mécanisme d'interrogation (minuterie d'événements setInterval) qui appelle périodiquement le script côté serveur pour vérifier la disponibilité d'un autre fichier à télécharger.

Lorsque le script côté serveur est exécuté s'il détecte un fichier de lecteur correspondant aux critères de recherche *, il renvoie l'URL du script lui-même ajouté aux paramètres:

? servefile = true & id = le_id_of_the_google_drive_file

(* Le critère de recherche dans cet exemple simple est codé en dur dans le script côté serveur. Il peut facilement être transmis du client au serveur si nécessaire.)

Ces informations sont renvoyées sous forme de chaîne au client via le mécanisme reconnu avec SuccèsHandler.

Le script Java client met alors à jour le HREF d'une ancre masquée avec ces informations renvoyées, puis clique automatiquement sur l'ancre.

Cela provoque un autre appel de l'application / script à lancer. Lorsque la nouvelle invocation de l'application lance le doGet, il détecte le paramètre servefile et au lieu de renvoyer l'interface utilisateur, il renvoie le fichier au navigateur. Le fichier renvoyé sera celui identifié par le paramètre ID fourni, précédemment renvoyé par la recherche décrite ci-dessus.

Étant donné que le fichier avec l'ID fourni existe toujours, il sera téléchargé et le nouvel appel de l'application se fermera, laissant la première invocation pour répéter ce processus.

Un bouton est fourni sur l’interface simple si l’utilisateur / le testeur est impatient d’attendre le temporisateur, mais ce n’est pas obligatoire et peut être supprimé.

Le formulaire simple peut bien sûr être étendu pour fournir une interface utilisateur plus riche si nécessaire. Comme fournir les critères de recherche de fichier.

Lire Script Google App Web à télécharger automatiquement à partir de Google Drive en ligne: https://riptutorial.com/fr/google-apps-script/topic/8212/script-google-app-web-a-telecharger-automatiquement-a-partir-de-google-drive

Chapitre 14: Service de tableur

Remarques

La référence API officielle pour le service Spreadsheet est disponible à l' adresse https://developers.google.com/apps-script/reference/spreadsheet/ .

Examples

Drap

Obtenir une référence à un onglet de feuille nommée

```
var spread_sheet = SpreadsheetApp.getActiveSpreadsheet();//Get active spreadsheet
var sheet_with_name_a = spread_sheet.getSheetByName("sheet_tab_name");
```

Obtenir un onglet actif

```
var spread_sheet = SpreadsheetApp.getActiveSpreadsheet();
var active_sheet = spread_sheet.getActiveSheet();
```

Insérer une colonne

```
var spread_sheet = SpreadsheetApp.getActiveSpreadsheet();
var active_sheet = spread_sheet.getActiveSheet();
active_sheet.insertColumnAfter(1);  // This inserts a column after the first column position
active_sheet.insertColumnBefore(1);  // This inserts a column in the first column position
active_sheet.insertColumns(1);  // Shifts all columns by one
active_sheet.insertColumns(1, 3);  // Shifts all columns by three
active_sheet.insertColumnsAfter(1);  // This inserts a column in the second column position
active_sheet.insertColumnsBefore(1, 5);  // This inserts five columns before the first column
```

Insérer une ligne

```
var spread_sheet = SpreadsheetApp.getActiveSpreadsheet();
var active_sheet = spread_sheet.getActiveSheet();
active_sheet.insertRowAfter(1);  // This inserts a row after the first row position
active_sheet.insertRowBefore(1);  // This inserts a row in the first row position
active_sheet.insertRows(1);  // Shifts all rows by one
active_sheet.insertRows(1, 3);  // Shifts all rows by three
active_sheet.insertRowsAfter(1);  // This inserts a row in the second row position
active_sheet.insertRowsBefore(1, 5);  // This inserts five rows before the first row
```

Valeur de cellule

```
var spread_sheet = SpreadsheetApp.getActiveSpreadsheet();
var active_sheet = spread_sheet.getActiveSheet();
var cell = range.getCell(1, 1);
var cell_value = cell.getValue();
```

```
cell.setValue(100);
```

Copier des cellules

```
var spread_sheet = SpreadsheetApp.getActiveSpreadsheet();
var active_sheet = spread_sheet.getActiveSheet();
var rangeToCopy = active_sheet.getRange(1, 1, sheet.getMaxRows(), 5);
rangeToCopy.copyTo(sheet.getRange(1, 6));
```

Formule

```
var spread_sheet = SpreadsheetApp.getActiveSpreadsheet();
var active_sheet = spread_sheet.getActiveSheet();
var range = active_sheet.getRange("B5");
var formula = range.getFormula()
range.setFormula("=SUM(B3:B4)");
```

Copier une valeur d'une feuille dans la feuille courante

Imaginez que nous ayons une feuille de calcul Google distincte, et que nous avons besoin de la valeur de cellule B2 dans la cellule D5 de votre feuille actuelle.

```
function copyValueandPaste()
{
    var source = SpreadsheetApp.openById('spread sheet id is here'); //Separate spreadsheet
book
    var sourcesheet = source.getSheetByName('Sheet1'); //Sheet tab with source data
    var sourceCellValue = sourcesheet.getRange('B2').getValue(); // get B2 cell value

    var thisBook = SpreadsheetApp.getActive(); // Active spreadsheet book
    var thisSheet = thisBook.getSheetByName('Sheet1'); // Target sheet
    thisSheet.getRange('D5').setValue(sourceCellValue); //Set value to target sheet D5 cell
}
```

Vous pouvez trouver l'ID de feuille de calcul à partir de votre URL.

Récupère la dernière ligne dans une seule colonne

```
function lastRowForColumn(sheet, column){
  // Get the last row with data for the whole sheet.
  var numRows = sheet.getLastRow();

  // Get all data for the given column
  var data = sheet.getRange(1, column, numRows).getValues();

  // Iterate backwards and find first non empty cell
  for(var i = data.length - 1 ; i >= 0 ; i--){
    if (data[i][0] != null && data[i][0] != ""){
      return i + 1;
    }
  }
}
```

Insertion de tableaux en tant que lignes

L'insertion d'une ligne au bas d'une feuille de calcul est facile:

```
var someSheet = SpreadsheetApp.getActiveSpreadsheet().getSheets()[0];
someSheet.appendRow(["Frodo", "Baggins", "Hobbit", "The Shire", 33]);
```

Notez que cela va ajouter la ligne après la dernière ligne *non vide* .

Insérer une ligne quelque part au milieu est un peu plus compliqué:

```
var someSheet = SpreadsheetApp.getActiveSpreadsheet().getSheets()[0];

var newRowIndex = 2;
var row = ["Gandalf", "?", "Wizard", "?", 2019];
someSheet.insertRowBefore(newRowIndex);
// getRange(row, col, numRows, numCols)
someSheet.getRange(newRowIndex, 1, 1, row.length).setValues([row]); // Note 2D array!
```

Une grande partie de ce code inutile peut être extrait dans une fonction d'assistance:

```
function insertRowBefore(sheet, rowIndex, rowData) {
  sheet.insertRowBefore(rowIndex);
  sheet.getRange(rowIndex, 1, 1, rowData.length).setValues([rowData]);
}
```

Ce qui réduit notre exemple à juste:

```
var someSheet = SpreadsheetApp.getActiveSpreadsheet().getSheets()[0];
insertRowBefore(someSheet, 2, ["Gandalf", "?", "Wizard", "?", 2019]);
```

Lire Service de tableur en ligne: https://riptutorial.com/fr/google-apps-script/topic/2688/service-de-tableur

Chapitre 15: Service DriveApp

Remarques

Les types Google Mime ne peuvent pas être utilisés pour le troisième paramètre de types Mime. L'utilisation d'un type MIME Google entraînera une erreur indiquant:

Impossible d'utiliser "DriveApp.createFile ()" pour créer des types Google MIME. Veuillez utiliser Advanced Drive Service

MimeType.GOOGLE_APPS_SCRIPT

MimeType.GOOGLE_DOCS

MimeType.GOOGLE_DRAWINGS

MimeType.GOOGLE_FORMS

MimeType.GOOGLE_SHEETS

MimeType.GOOGLE_SLIDES

Examples

Créer un nouveau dossier dans le lecteur racine Google

```
function createNewFolderInGoogleDrive() {
  var folderName,newFolder;//Declare variable names

  folderName = "Test Folder " + new Date().toString().slice(0,15);//Create a new folder name
with date on end
  newFolder = DriveApp.createFolder(folderName);//Create a new folder in the root drive
};
```

Créer un nouveau fichier dans Google Drive d'un certain type Mime

```
function createGoogleDriveFileOfMimeType() {
  var content,fileName,newFile;//Declare variable names

  fileName = "Test File " + new Date().toString().slice(0,15);//Create a new file name with
date on end
  content = "This is the file Content";

  newFile = DriveApp.createFile(fileName,content,MimeType.JAVASCRIPT);//Create a new file in
the root folder
};
```

Créer un nouveau fichier texte dans le dossier du lecteur racine Google

```
function createGoogleDriveTextFile() {
  var content,fileName,newFile;//Declare variable names

  fileName = "Test Doc " + new Date().toString().slice(0,15);//Create a new file name with
date on end
  content = "This is the file Content";

  newFile = DriveApp.createFile(fileName,content);//Create a new text file in the root folder
};
```

Créer un nouveau fichier dans Google Drive à partir d'un blob

```
function createGoogleDriveFileWithBlob() {
  var blob,character,data,fileName,i,L,max,min,newFile,randomNmbr;//Declare variable names

  fileName = "Test Blob " + new Date().toString().slice(0,15);//Create a new file name with
date on end

  L = 500;//Define how many times to loop
  data = "";
  max = 126;
  min = 55;

  for (i=0;i<L;i+=1) {//Loop to create data
    randomNmbr = Math.floor(Math.random()*(max-min+1)+min);//Create a random number
    //Logger.log('randomNmbr: ' + randomNmbr);
    character = String.fromCharCode(randomNmbr);

    //Logger.log('character: ' + character);//Print the character to the Logs
    data = data + character;
  };

  blob = Utilities.newBlob(data, MimeType.PLAIN_TEXT, fileName);//Create a blob with random
characters

  newFile = DriveApp.createFile(blob);//Create a new file from a blob

  newFile.setName(fileName);//Set the file name of the new file
};
```

Récupère tous les dossiers - place les dossiers dans un jeton de continuation - puis récupère les jetons

```
function processGoogleDriveFolders() {
  var arrayAllFolderNames,continuationToken,folders,foldersFromToken,thisFolder;//Declare
variable names

  arrayAllFolderNames = [];//Create an empty array and assign it to this variable name

  folders = DriveApp.getFolders();//Get all folders from Google Drive in this account
  continuationToken = folders.getContinuationToken();//Get the continuation token

  Utilities.sleep(18000);//Pause the code for 3 seconds

  foldersFromToken = DriveApp.continueFolderIterator(continuationToken);//Get the original
folders stored in the token
  folders = null;//Delete the folders that were stored in the original variable, to prove that
```

```
the continuation token is working

  while (foldersFromToken.hasNext()) {//If there is a next folder, then continue looping
    thisFolder = foldersFromToken.next();//Get the next folder
    arrayAllFolderNames.push(thisFolder.getName());//Get the name of the next folder
  };

  Logger.log(arrayAllFolderNames);//print the folder names to the Logs
};
```

Obtenez tous les fichiers - mettez-les dans un jeton de continuation - puis récupérez-les

```
function processGoogleDriveFiles() {
  var arrayAllFileNames,continuationToken,files,filesFromToken,fileIterator,thisFile;//Declare
variable names

  arrayAllFileNames = [];//Create an empty array and assign it to this variable name

  files = DriveApp.getFiles();//Get all files from Google Drive in this account
  continuationToken = files.getContinuationToken();//Get the continuation token

  Utilities.sleep(18000);//Pause the code for 3 seconds

  filesFromToken = DriveApp.continueFileIterator(continuationToken);//Get the original files
stored in the token
  files = null;//Delete the files that were stored in the original variable, to prove that the
continuation token is working

  while (filesFromToken.hasNext()) {//If there is a next file, then continue looping
    thisFile = filesFromToken.next();//Get the next file
    arrayAllFileNames.push(thisFile.getName());//Get the name of the next file
  };

  Logger.log(arrayAllFileNames);
};
```

Lire Service DriveApp en ligne: https://riptutorial.com/fr/google-apps-script/topic/6395/service-driveapp

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec google-apps-script	Albert Portnoy, Community, Douglas Gaskell, iJay, MShoaib91, Rubén, Saloni Vithalani, Shyam Kansagra, Spencer Easton, sudo bangbang, Supertopoz
2	Appels clients à Google apps-script	Supertopoz
3	Apps Script Web Apps	Douglas Gaskell
4	Créer une fonction personnalisée pour Google Sheets	Francky_V, Joshua Dawson, Pierre-Marie Richard, Rubén
5	DriveApp	Brian, Kos, nibarius, Sandy Good, Wolfgang
6	DriveApp - getFileById (id)	Sandy Good
7	DriveApp Service - Fichiers par type et chaîne de recherche	nibarius, Sandy Good
8	Feuille active SpreadsheetApp	iJay
9	Feuille de calcul Ajouter un menu	Bishal, iJay, nibarius
10	Firebase et AppScript: Introduction	Joseba, Vishal Vishwakarma
11	GmailApp	nibarius
12	Google feuilles MailApp	Bhupendra Piprava, Brian, Jordan Rhea, Kos, nibarius, Saloni Vithalani
13	Script Google App Web à télécharger automatiquement à partir de Google	Walter

	Drive	
14	Service de tableur	cdrini, iJay, nibarius, Sandy Good, sudo bangbang
15	Service DriveApp	Sandy Good