EBook Gratuito

APPRENDIMENTO

google-apps-script

Free unaffiliated eBook created from
Stack Overflow contributors.

#google-apps-script

Sommario

Di

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

Capitolo 1: Iniziare con google-apps-script

Osservazioni

La panoramica ufficiale di Google Apps Script è pubblicata all'indirizzo http://www.google.com/script/start , da lì

> Google Apps Script è un linguaggio di scripting cloud JavaScript che fornisce semplici metodi per automatizzare le attività tra prodotti Google e servizi di terze parti e creare applicazioni web.

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

> Apps Script è basato su JavaScript 1.6 , oltre a alcune funzionalità da 1.7 e 1.8 . Sono quindi disponibili molte funzioni JavaScript di base oltre ai servizi Google integrati e avanzati : è possibile utilizzare oggetti comuni come Array , Date , RegExp e così via , nonché gli oggetti globali Math e Object . Tuttavia, poiché il codice Apps Script viene eseguito sui server di Google (non sul lato client, ad eccezione delle pagine del servizio HTML), le funzionalità basate su browser come la manipolazione DOM o l'API Window non sono disponibili.

Examples

Installazione o configurazione

Lo script di Google Apps non richiede installazione o installazione. L'unico requisito è un account Google. Un account Gmail funziona come un account Google Apps for Work / Education / Government. Puoi creare un nuovo account Google accedendo a accounts.google.com

Inizia il tuo primo script andando su script.google.com . Puoi anche accedere a Google Apps Script sotto gli `tools -> Script editor...` di molte Google Apps, ad es. *Documenti, Fogli, Moduli ecc* . Lo script di Google Apps può anche essere aggiunto direttamente a Google Drive con la funzione `Connect more apps..`

La documentazione ufficiale è disponibile all'indirizzo developers.google.com/apps- script / .

Per gli script delle app da eseguire, devono contenere un file code.gs. Il file code.gs deve contenere una funzione denominata doGet (script autonomi) o una funzione onOpen (script di aggiunta). Le partenze rapide nella documentazione contengono esempi.

Se un'API è attiva nello script app, deve essere attivata anche nella console degli sviluppatori. Tuttavia, la console degli sviluppatori contiene le API che possono essere attivate ma che non compaiono nell'interfaccia dell'app-script. Ad esempio, l'SDK di Marketplace deve essere attivato nella console degli sviluppatori prima che l'app possa essere pubblicata nel Play Store di Google o in una distribuzione di dominio G suite.

Per le app Google per istruzione / lavoro / amministrazione esistono impostazioni nella console di amministrazione del dominio che possono essere regolate per consentire o impedire l'esecuzione degli script app.

Tipi di script

Gli script di Google App sono di tre tipi.

- Indipendente, autonomo
- Associato a Google Apps
- App Web

Script autonomo

Gli script autonomi non sono associati a nessuna app Google, ad esempio *documenti, fogli o moduli, ecc.* È possibile creare script standalone visitando *script.google.com* o collegando lo script app Google con l'unità Google. Lo script standalone può essere utilizzato per programmare le app di Google in modo indipendente, può essere utilizzato come app Web o può essere configurato per essere eseguito automaticamente da un trigger installabile. Vedi la documentazione per lo script standalone.

Associato a Google Apps

Script associato a Google Apps noto anche come script associato al contenitore; a differenza degli script standalone, sono associati alle app Google, ad esempio *Google Documenti o Fogli Google, ecc. Lo* script rilegato contenitore può essere creato selezionando `tools> Script editor` dall'app Google. Alcune funzioni come le finestre di dialogo, i prompt, i menu e la barra laterale sono fornite solo da script vincolati al contenitore. Inoltre, lo script container-bound viene utilizzato per creare componenti aggiuntivi di Google . Vedere la documentazione per gli script rilegati dal contenitore.

App Web

Lo script per app di Google può essere utilizzato come app web in quanto accessibili dal browser. L'app Web può fornire l'interfaccia utente sul browser e può utilizzare app google come *documenti, fogli, ecc.* Sia gli script autonomi che gli script associati a Google Apps possono essere trasformati in app web. Affinché qualsiasi script funzioni come app Web, lo script deve soddisfare due requisiti:

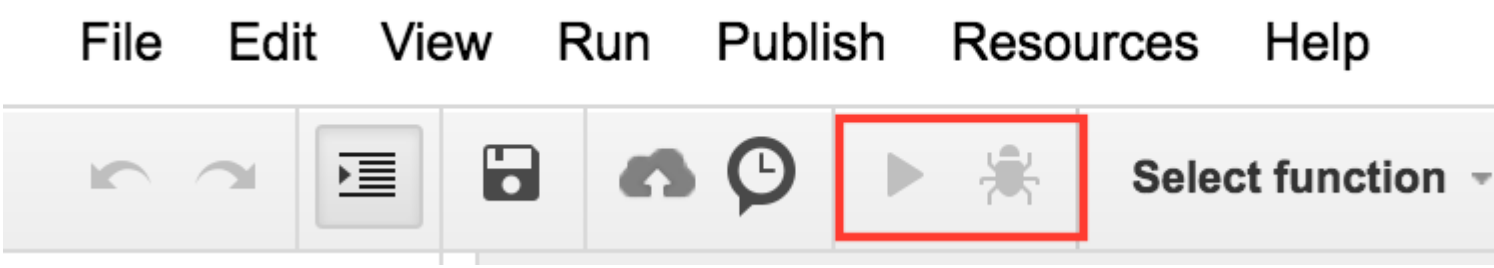
- include una funzione `doGet()` o `doPost()` .
- La funzione restituisce un oggetto HTML HtmlOutput o un oggetto TextOutput del servizio contenuto.

`doGet()` , le `doGet()` e `doPost()` funzionano come http get e post handler di richiesta rispettivamente.

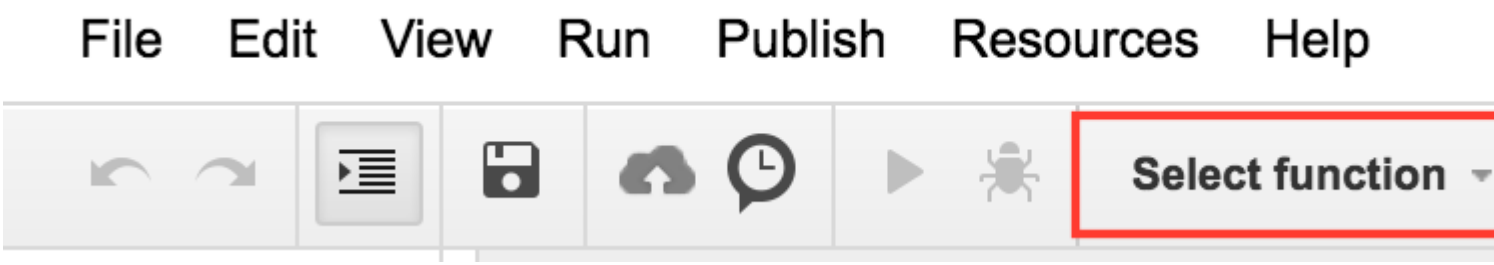
Per maggiori dettagli su Web Apps, consultare la documentazione ufficiale.

Esecuzione / debug del tuo script

Prova a eseguire il codice dalla barra degli strumenti come mostrato di seguito:



Nel tuo codice, se hai più di una funzione, prima di eseguirlo dovresti menzionare la funzione che vuoi eseguire. Per esempio :



In alternativa, puoi premere **ctrl + r** dalla tastiera per eseguire il codice. Prima salverà il codice, se non è stato salvato, quindi lo eseguirà. Ma, affinché funzioni, è necessario aver selezionato la funzione, come si vede nell'immagine sopra.

Inoltre, se lo script viene chiamato da alcune attività esterne, potrai comunque visualizzare i registri facendo clic su Visualizza-> Registra se stai loggando qualcosa dopo che il codice è stato eseguito.

Ciao mondo

Stiamo per dire Ciao come una finestra di messaggio.

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

Per eseguire lo script, fare clic su ▶ o selezionare la voce di menu **Esegui** -> **helloWorld**

Uno sguardo più approfondito a Google Apps Script

Google Apps Script è una piattaforma basata su JavaScript come servizio utilizzato principalmente per automatizzare ed estendere Google Apps. Apps Script viene eseguito esclusivamente sull'infrastruttura di Google che non richiede provisioning o configurazione del server. Un IDE online funge da interfaccia per l'intera piattaforma che collega tutti i servizi disponibili per Apps Script. L'autenticazione dell'utente viene inserita nella piattaforma tramite OAuth2 e non richiede codice o impostazione dall'autore dello script.

Apps Script viene eseguito sul lato server, ma può avere interfacce utente create con Html, CSS, JavaScript o qualsiasi altra tecnologia supportata dal browser. A differenza di Nodejs, che è guidato dagli eventi, gli script di app vengono eseguiti in un modello con thread. Tutte le chiamate a uno script generano un'istanza univoca di quello script che viene eseguito indipendentemente da tutte le altre istanze. Quando un'istanza di uno script termina l'esecuzione viene distrutta.

Le funzioni in Apps Script stanno bloccando in modo tale che i pattern di callback e asincrono non siano necessari. Il blocco viene utilizzato per impedire che sezioni critiche del codice, come il file I / O, vengano eseguite simultaneamente da istanze diverse.

In pratica scrivere Apps Scripts è semplice. Di seguito è riportato un semplice script che crea un nuovo foglio di calcolo da un foglio di calcolo del modello.

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

Leggi Iniziare con google-apps-script online: https://riptutorial.com/it/google-apps-script/topic/1154/iniziare-con-google-apps-script

Capitolo 2: Aggiungi foglio di calcolo

Sintassi

1. addMenu (nome, sottomenu)

Parametri

Nome	Descrizione
nome	il nome del menu da creare
sottomenu	una serie di mappe JavaScript

Osservazioni

In genere, è necessario chiamare addMenu dalla funzione onOpen in modo che il menu venga creato automaticamente quando viene caricato il foglio di calcolo.

```
 // 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

Crea un nuovo menu

Crea un nuovo menu nell'interfaccia utente del foglio di calcolo. Ogni voce di menu esegue una funzione definita dall'utente.

```
   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);
```

Crea menu personalizzato

/ *

Metodo: Creare un menu personalizzato Questa è la prima funzione da chiamare quando si caricano le app

---

* /

```
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();
}
```

Leggi Aggiungi foglio di calcolo online: https://riptutorial.com/it/google-apps-script/topic/4253/aggiungi-foglio-di-calcolo

Capitolo 3: App Script Web App

Osservazioni

Questa è un'app web di esempio, il bit sul lato client mostra alcuni elementi di progettazione UX di base, come un pulsante di invio disabilitato quando il modulo viene inviato, o un messaggio di errore se fallisce ... ecc.

Il bit di Apps Script è molto semplice. Contiene solo il codice necessario per servire l'html e per convalidare il campo.

Ecco un link a questa app di esempio in azione: Modulo di script app di esempio

Nota: è necessario aver effettuato l'accesso a un account Google.

La struttura del file Apps Script è così:

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

Examples

Modulo Web App

Script delle app:

```
//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>
```

Leggi App Script Web App online: https://riptutorial.com/it/google-apps-script/topic/4874/app-script-web-app

Capitolo 4: Client chiama a Google apps-script

introduzione

L'appscript di Google funziona bene come piattaforma stand alone e nel formato di aggiunta per documenti, fogli e moduli Google. Tuttavia, ci sono momenti in cui un browser client potrebbe dover chiamare una app di Google per eseguire qualche azione.

Pertanto, Google ha introdotto le richieste lato client agli script delle app di Google. Per risolvere questo problema, Google ha introdotto le librerie lato client

Examples

Questo è un esempio di una chiamata lato client a uno script di app di 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>
```

Leggi Client chiama a Google apps-script online: https://riptutorial.com/it/google-apps-script/topic/8875/client-chiama-a-google-apps-script

Capitolo 5: Crea una funzione personalizzata per Fogli Google

introduzione

Una funzione personalizzata in Google Documenti è legata a un documento specifico (e quindi può essere utilizzata solo in quel documento).

Deve quindi essere creato con la modifica dello script di quel documento (Strumenti -> Script Editor). Una volta salvato, può quindi essere utilizzato come qualsiasi altra formula di foglio di calcolo normale.

Examples

Costante personalizzata di gravità standard

Questa funzione restituisce la costante di gravità standard nelle unità di accelerazione specificate (1 per cm / s², 2 per ft / s², 3 per 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;
}
```

Per utilizzare la funzione, è necessario essere associato a un foglio di calcolo utilizzando l'editor di script (Strumenti -> Editor di script ...). Una volta aggiunta la funzione, può essere utilizzata come qualsiasi altra funzione di Google fogli chiamando la funzione nella formula di una cella.

Nota come la funzione si presenta in completamento automatico quando viene digitata in una

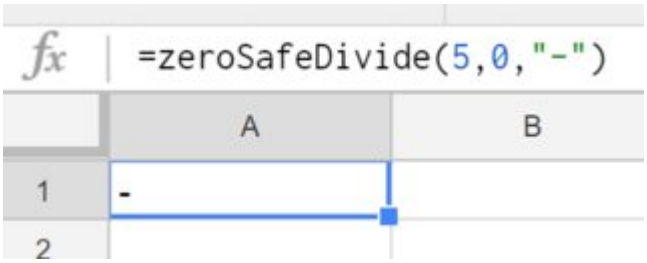
formula. Ciò è dovuto al commento a più righe sopra la dichiarazione di funzione che viene utilizzata per descrivere ciò che la funzione fa in modo simile a JSDoc e Javadoc. Per visualizzare la formula in completamento automatico, il tag @customfunction deve essere specificato nel commento.

Esempio di base

Per evitare errori `#DIV/0` sgradevoli in un foglio di calcolo, è possibile utilizzare una funzione personalizzata.

```
/**
 * 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;
}
```

Per utilizzare la funzione, è necessario essere associato a un foglio di calcolo utilizzando l'editor di script (**Strumenti -> Editor di script ...**). Una volta aggiunta la funzione, può essere utilizzata come qualsiasi altra funzione di Google fogli chiamando la funzione nella formula di una cella.



Nota come la funzione si presenta in completamento automatico quando viene digitata in una formula. Ciò è dovuto al commento a più righe sopra la dichiarazione di funzione che viene utilizzata per descrivere ciò che la funzione fa in modo simile a JSDoc e Javadoc. Per visualizzare la formula in completamento automatico, il tag `@customfunction` deve essere specificato nel commento.

Leggi Crea una funzione personalizzata per Fogli Google online: https://riptutorial.com/it/google-apps-script/topic/5572/crea-una-funzione-personalizzata-per-fogli-google

Capitolo 6: DriveApp

Examples

Crea una nuova cartella in una radice di Google Drive

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

Utilizza la funzione `createNewFolderInGoogleDrive` per creare una cartella denominata `Test folder` in una radice di Google Drive:

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

`newFolder` ha un tipo di cartella di classe :

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

Crea un nuovo file in Google Drive di un certo tipo 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
};
```

Crea un nuovo file di testo nella cartella principale di 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
};
```

Crea un nuovo file nell'unità Google da 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
};
```

Ottieni tutte le cartelle - inserisci le cartelle in un token di continuazione - quindi recupera dal token

```
 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
};
```

Ottieni tutti i file - inseriscili in un token di continuazione, quindi recuperali

```
 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);
};
```

Aggiungi una cartella all'unità principale

```
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
};
```

Crea un nuovo file di testo e aggiungilo alla cartella principale

```
 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
};
```

Ottieni tutti i file in una cartella Drive

```
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++;
  }
}
```

Leggi DriveApp online: https://riptutorial.com/it/google-apps-script/topic/5363/driveapp

Capitolo 7: DriveApp - getFileById (id)

Osservazioni

È anche possibile ottenere un file tramite l'URL del file. L'ID di un file è nell'URL, quindi l'utilizzo dell'ID anziché dell'intero URL indica che il parametro è più breve. La memorizzazione dell'URL anziché dell'ID occupa più spazio.

Examples

Ottieni un file da Google Drive utilizzando l'ID del file

```
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
};
```

Leggi DriveApp - getFileById (id) online: https://riptutorial.com/it/google-apps-script/topic/6087/driveapp---getfilebyid--id-

Capitolo 8: Firebase e AppScript: Introduzione

introduzione

Integrare Firebase con Google AppScript per leggere e scrivere dati nel database Firebase.

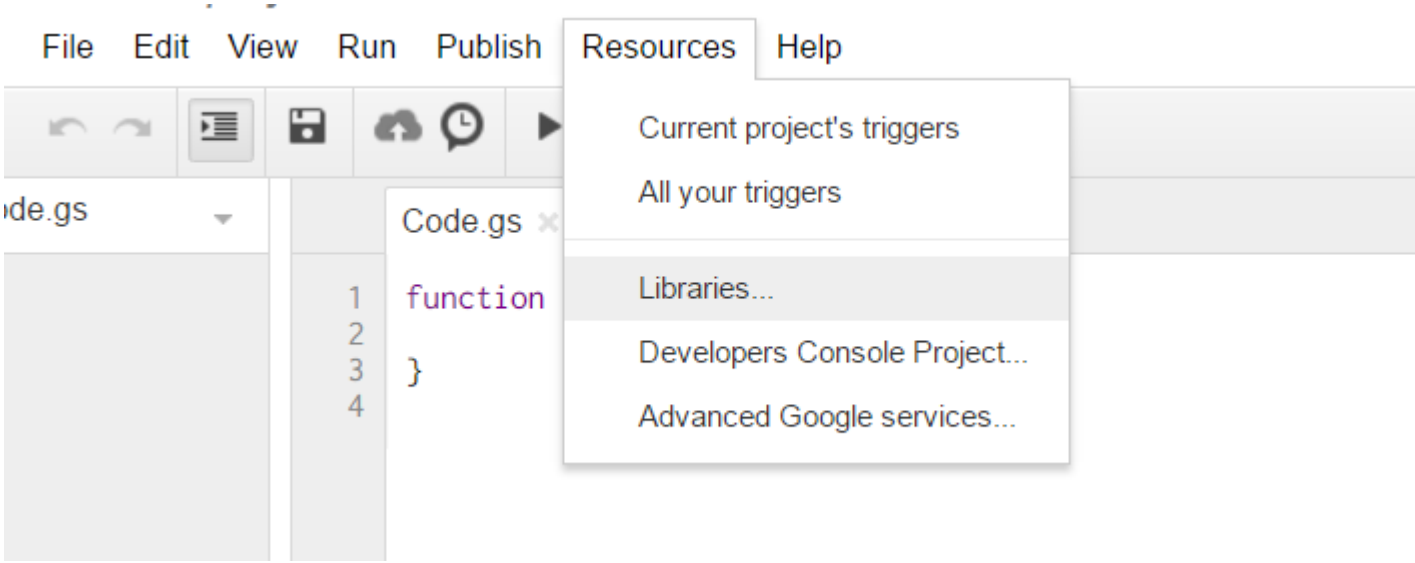
Firebase è un sistema di database NoSQL di Google che utilizza database in tempo reale per aiutare a creare e ospitare applicazioni su dispositivi mobili, desktop e tablet. I database NoSQL utilizzano gli oggetti JSON per archiviare i dati in formato strutturato.

Examples

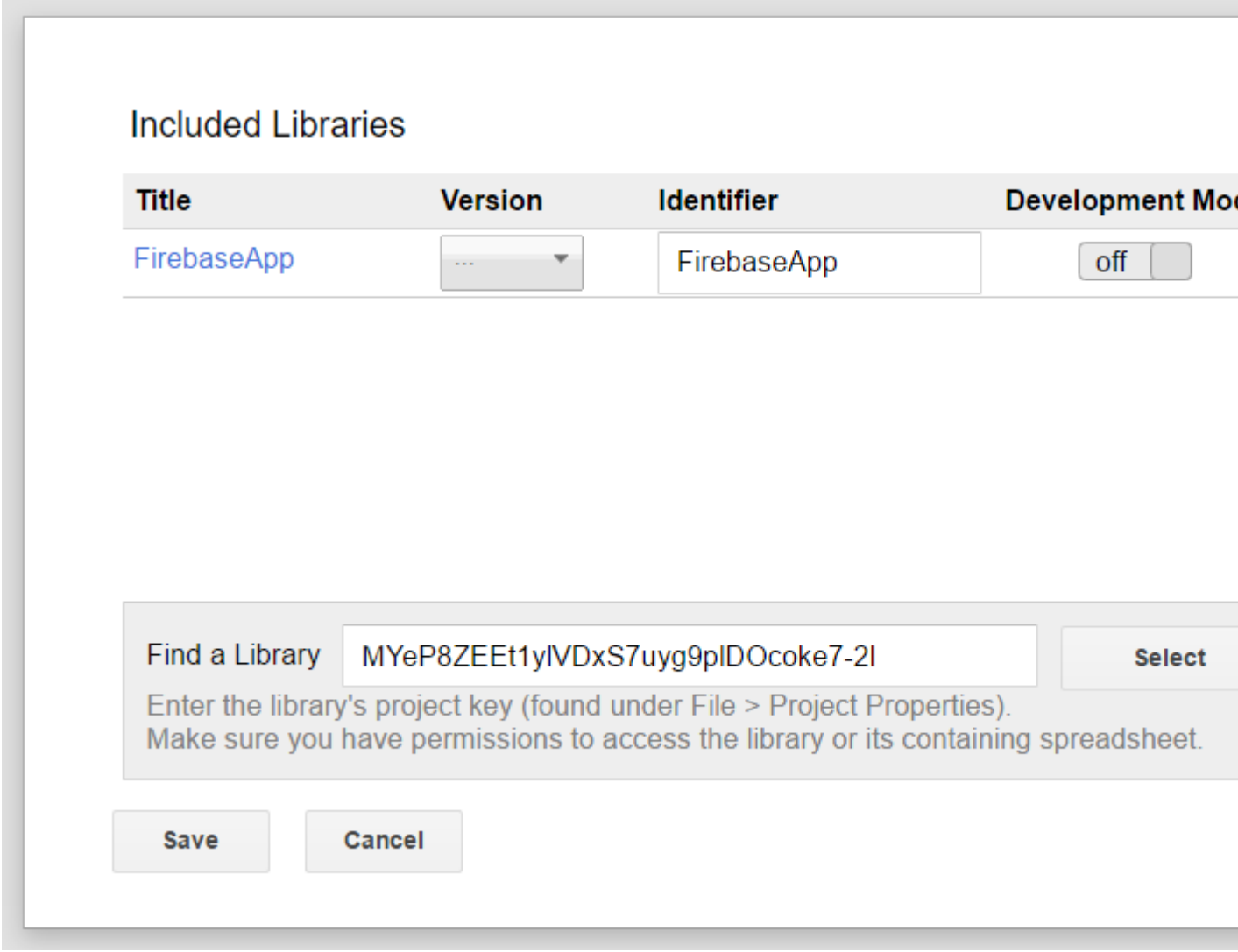
Connessione a un progetto Firebase in GAS e trasferimento di dati da Google Spreadsheet a Firebase

Installa la risorsa Firebase in AppScript

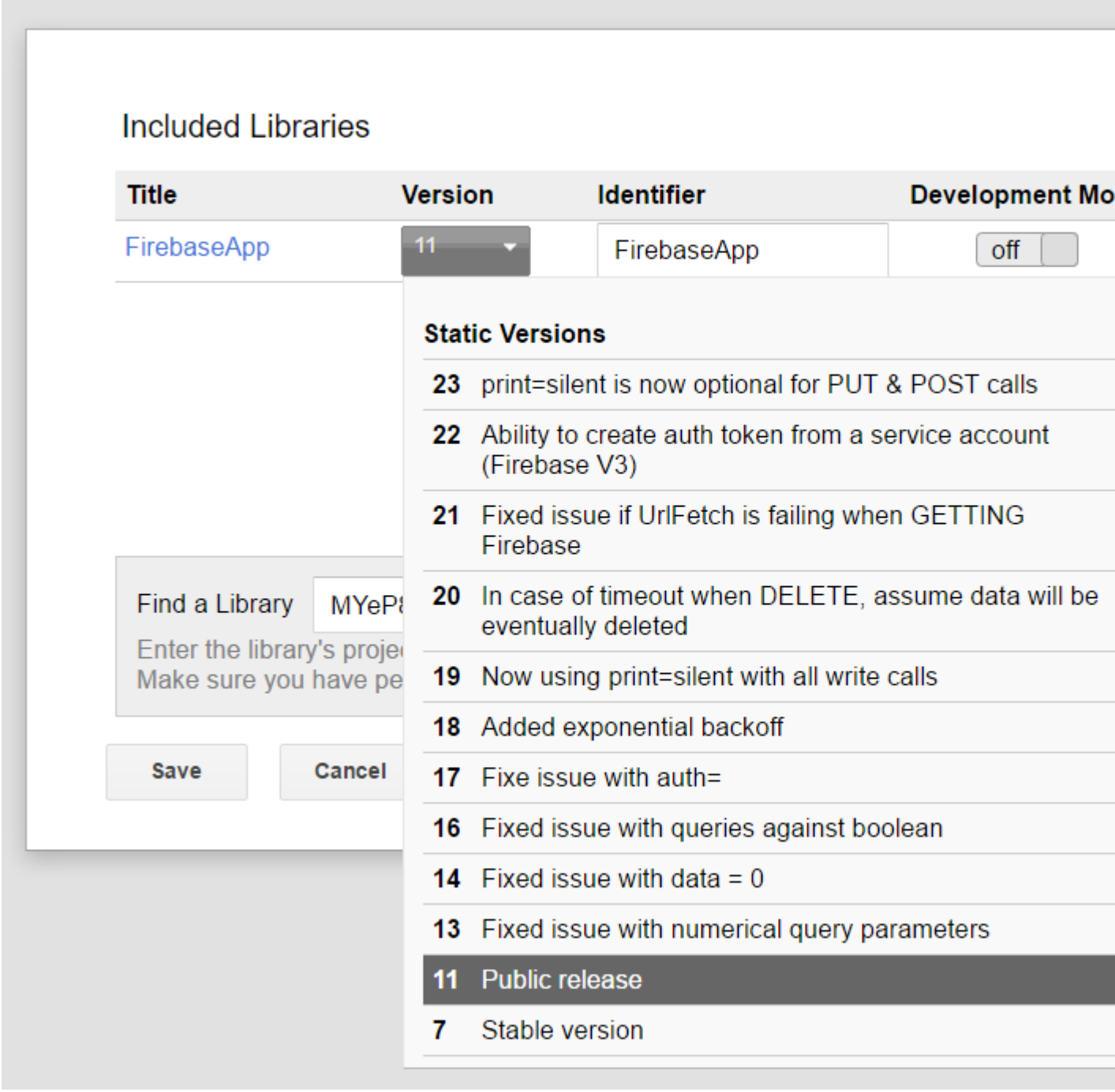
- Per farlo clicca su Risorse e poi su Librerie.
- Firebase ha una chiave di libreria di progetto univoca che deve essere installata in AppScript.



- Fare clic su Librerie Viene visualizzato il seguente popup. Inserisci la seguente chiave di progetto nella casella di testo. **MYeP8ZEEt1ylVDxS7uyg9plDOcoke7-2l** Questa è la chiave di libreria del progetto per Firebase.



- Ora nella versione scegli la versione pubblica stabile.



- Clicca su Salva. Ora Firebase è stato installato con successo in AppScript affinché tu possa lavorare.

Ora facciamo un esempio per leggere e scrivere dati da Firebase.

- Ora prendiamo una tabella di esempio progettata in Fogli Google.

	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

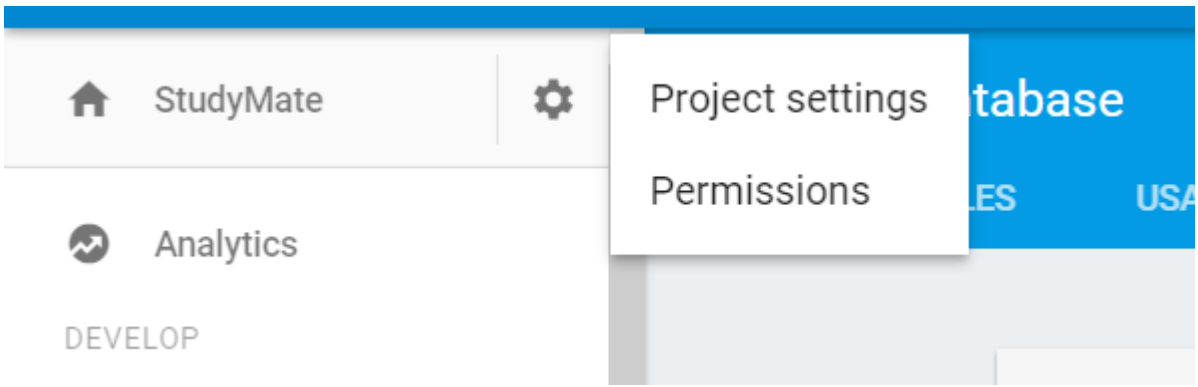
- Ora per costruire il database nel Firebase usando questa tabella nei fogli. Aggiungi il seguente codice in AppScript.

```
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);
}
```

Sostituisci l'ID del foglio di calcolo e il firebaseURL e la chiave segreta.

Come trovare firebaseURL e la chiave segreta?

- Vai alla Firebase Dashboard e fai clic su impostazioni gear nell'angolo in alto a sinistra. Clicca su Impostazioni progetto.



- Vai alla sezione Account di servizio puoi trovare databaseURL. Questo serve come firebaseURL.
- Ora fai clic sulla scheda Database Secrets e puoi trovare la chiave segreta.

Ora hai inserito firebaseURL e la chiave segreta. Ora sei pronto per partire. Fare clic su Esegui codice nel motore AppScript.

- Richiederà di rivedere i permessi la prima volta quando corri.
- Fai clic su Verifica autorizzazioni e Consenti.
- Ora esegui la tua funzione e puoi vedere la tabella creata nel Database Firebase.

Per vedere il database vai al dashboard di Firebase e clicca sul Database per visualizzare il database.

Altre funzioni per implementare leggere e scrivere.

1. Scrivere un semplice dato per verificare se la connessione funziona o meno.

```
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. Leggere tutti i dati

```
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);
          }
}
```

I dati letti sono visualizzati nei registri. Per controllare i registri fare clic su fare clic su Visualizza → Registri o semplicemente utilizzare Control + Invio.

3. Leggere un record specifico

```
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);
}
```

I dati letti sono visualizzati nei registri. Per controllare i registri fare clic su fare clic su Visualizza → Registri o semplicemente utilizzare Control + Invio.

4. Per aggiornare un record specifico.

```
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");
}
```

Leggi Firebase e AppScript: Introduzione online: https://riptutorial.com/it/google-apps-script/topic/9417/firebase-e-appscript--introduzione

Capitolo 9: Foglio attivo di SpreadsheetApp

Osservazioni

Metodo: getActive ()

Tipo di reso : foglio di calcolo

Examples

getActive () - Ottieni foglio di calcolo attivo

Questo restituisce il foglio di calcolo attivo, o null se non ce n'è.

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

Leggi Foglio attivo di SpreadsheetApp online: https://riptutorial.com/it/google-apps-script/topic/5861/foglio-attivo-di-spreadsheetapp

Capitolo 10: GmailApp

Osservazioni

Consulta anche il riferimento API ufficiale per GmailApp per ulteriori dettagli sui metodi disponibili.

Examples

Ottieni un file CSV allegato a una posta

Supponiamo che abbiamo un sistema che invia report giornalieri via e-mail sotto forma di file CSV allegati e che vogliamo accedervi.

```
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;
}
```

Leggi GmailApp online: https://riptutorial.com/it/google-apps-script/topic/5899/gmailapp

Capitolo 11: Google invia MailApp

introduzione

Questo servizio consente agli utenti di inviare e-mail con il controllo completo sul contenuto dell'e-mail. A differenza di GmailApp, l'unico scopo di MailApp è l'invio di e-mail. MailApp non può accedere alla posta in arrivo di Gmail di un utente.

Le modifiche agli script scritti usando GmailApp hanno più probabilità di innescare una richiesta di autorizzazione da un utente rispetto agli script MailApp.

Examples

Un esempio base di MailApp

MailApp è l'API di Google App Script che può essere utilizzata per inviare posta

```
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 è limitata alle quote basate sul tuo account Google:

- Utente consumatore (ad esempio, account Gmail personale): 100 destinatari / giorno
- Cliente di Google Apps (legacy): 100 destinatari / giorno
- GSuite (base / Gov / Edu / Business): 1500 destinatari / giorno

Puoi controllare la tua quota di posta elettronica all'interno di `MailApp`

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

Accedi ai dati dal foglio

```
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;
}
```

Puoi anche modificare la funzione di cui sopra come segue per ottenere un intervallo di dati dinamico dal contenuto presente nel foglio:

```
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;
}
```

Usa i dati del Foglio per inviare email

Dato - A hanno foglio di dipendenti che hanno richiesto il rimborso.

Requisito - Dovremmo inviare una email al dipendente quando viene elaborato il rimborso

Quindi, il foglio è il seguente:

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

La funzione per l'invio di una e-mail è la seguente:

```
function getDataSheet() {

  sheet = SpreadsheetApp.getActiveSheet();

  startRow = 2;  // First row of data to process
  numRows = 100;   // Number of rows to process
```

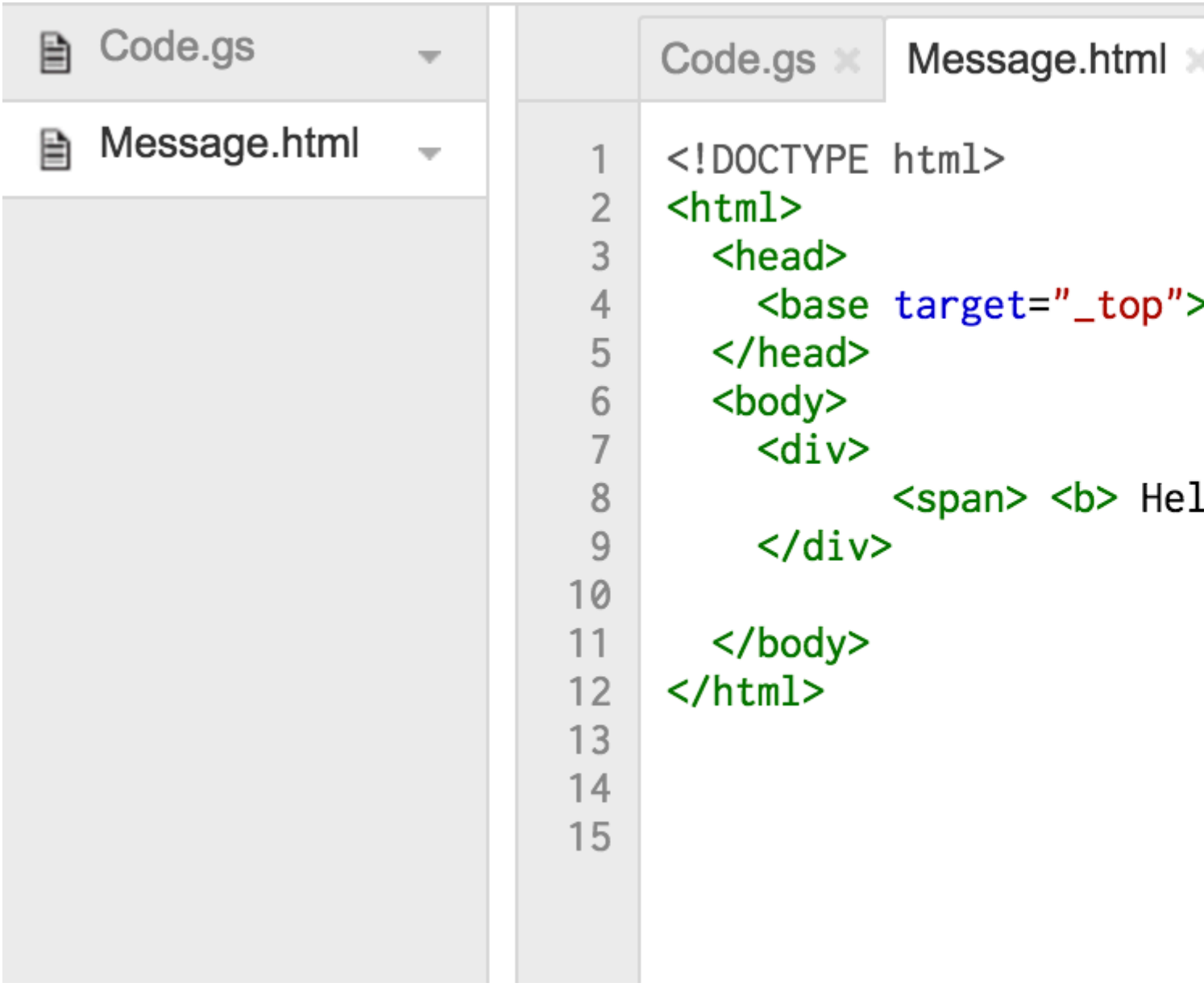
```
  startCol = 1;  //First column of data to process
  numCols = 15;    // Number of columns to process

  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

Invio di contenuto HTML nella posta

Nell'esempio precedente, se vogliamo inviare il contenuto HTML come messaggio nell'e-mail, quindi creare un file HTML andando su **File -> Nuovo -> File HTML**

Ora puoi vedere un file HTML oltre al tuo file gs come segue:



Ora, aggiorna il metodo *getMessage ()* dall'esempio precedente come segue:

```
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;
}
```

Anche la chiamata a *MailApp* api deve essere modificata

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

Quindi l'intero codice sarà il seguente:

```
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);
    }
  }
}
```

Leggi Google invia MailApp online: https://riptutorial.com/it/google-apps-script/topic/5298/google-invia-mailapp

Capitolo 12: Script di Google Web App per il download automatico da Google Drive

introduzione

Questo semplice Script Web di Google Apps (Standalone) consente a Google Drive di ripetere il polling dei file da scaricare sul PC locale dell'utente. Mostra come utilizzare uno script app per fornire la funzione di entrambi: 1. Interfaccia utente (una semplice in questo esempio) 2. La pagina di download del file. Per una spiegazione più completa di come funziona leggi l'esempio "Come funziona".

Osservazioni

Il Web Script deve essere pubblicato per funzionare.

I pop up devono essere abilitati per 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')
  }
}
```

Come funziona

Google Drive (Standalone) Web App per scaricare automaticamente (Poll) i file da Drive al PC locale dell'utente (cartella di download).

DriveApp fornisce meccanismi per la ricerca e il download di file. Tuttavia, il meccanismo di download presenta alcune limitazioni gravi a causa dell'architettura client / server ereditata da Google Apps. (Nessuna colpa di Google)

Il lato server DriveApp non fornisce una funzione diretta per il download sul PC locale perché il server non ha idea di dove si trova il client e il download del file sul server stesso sarebbe privo di significato.

Il codice lato server richiede un meccanismo per fornire al codice lato client i dati del file o un collegamento al file. Entrambi questi meccanismi sono forniti ma i dati del primo sono limitati a essere usati direttamente dal codice lato client. Il client non ha alcun meccanismo per salvare i dati, una volta ottenuti, sul PC locale. Quindi può essere usato per visualizzare i dati sulla pagina web stessa.

Il secondo meccanismo consente di rinviare l'url dello script (stesso) o l'url del file di Drive. L'url del file di Drive non è molto utile in quanto non può essere utilizzato direttamente nel browser client per scaricare il file. Inserendo questo url in ancora (e facendo clic su di esso) si ottiene solo una pagina Web che si apre ma in realtà non fa nulla (tranne eventualmente visualizzare il file online).

Questo lascia l'url dello script. Tuttavia l'url dello script fornisce solo lo script e non il file.

Per avviare un download, il file dal servizio Drive deve essere restituito dalla funzione doGet / doPost dello script lato server utilizzando ContentService createTextOutput esattamente come mostrato nelle guide online di Google. Tuttavia questo implica che non ci può essere nessun altro elemento dell'interfaccia utente sulla pagina web generata dai risultati restituiti da doGet / doPost.

Questo ci lascia con una soluzione molto poco attraente. Una pagina Web vuota senza elementi

utente dell'interfaccia utente che scarica una pagina, si chiude e richiede l'apertura manuale quando è richiesto un altro download.

Ovviamente un'altra pagina web di hosting potrebbe fornire l'interfaccia utente e il collegamento allo script di download di app Web per risolvere questo problema.

Questo script utilizza un approccio di Dr. Jekyll e Mr Hyde per risolvere questo problema.

Se lo script viene aperto senza parametri per GET (doGet), per impostazione predefinita viene visualizzato un modulo. Questa sarà la condizione in cui l'app pubblicata viene aperta per la prima volta da un utente. Il modulo fornito in questo esempio è estremamente semplice.

Se lo script viene aperto con il parametro servefile = true, lo script si comporta come download di file di Drive.

Il lato client javascript contiene un meccanismo di polling (timer evento setInterval) che periodicamente richiama lo script lato server per verificare la disponibilità di un altro file da scaricare.

Quando lo script lato server viene eseguito se trova qualsiasi file di Drive che corrisponde ai criteri di ricerca * restituisce l'URL dello script stesso aggiunto ai parametri:

? Servefile = true & id = the_id_of_the_google_drive_file

(* I criteri di ricerca in questo semplice esempio sono codificati nello script lato server e possono essere facilmente passati dal client al server se necessario).

Questa informazione viene restituita come stringa al client tramite il meccanismo riconosciuto withSuccessHandler.

Lo script java del client aggiorna l'HREF di un'ancora nascosta con queste informazioni restituite e quindi fa clic sull'ancora automaticamente.

Ciò causa l'avvio di un'altra chiamata dell'app / script. Quando la nuova chiamata dell'app viene avviata, doGet rileverà il parametro servefile e invece di restituire l'interfaccia utente restituirà il file al browser. Il file restituito sarà quello identificato dal parametro ID fornito precedentemente restituito dalla ricerca sopra descritta.

Dato che il file con l'ID fornito esiste ancora, verrà scaricato e la nuova chiamata dell'app si chiuderà lasciando la prima chiamata a ripetere questo processo.

Sull'interfaccia semplice viene fornito un pulsante se l'utente / tester diventa impaziente di attendere il timer ma non è necessario e può essere rimosso.

La semplice forma può naturalmente essere estesa per fornire un'interfaccia utente più ricca se necessario. Come fornire i criteri di ricerca del file.

Leggi Script di Google Web App per il download automatico da Google Drive online: https://riptutorial.com/it/google-apps-script/topic/8212/script-di-google-web-app-per-il-download-automatico-da-google-drive

Capitolo 13: Servizio di foglio di calcolo

Osservazioni

Il riferimento API ufficiale per il servizio Foglio di calcolo è disponibile all'indirizzo https://developers.google.com/apps-script/reference/spreadsheet/ .

Examples

Foglio

Ottenere un riferimento a una scheda di un foglio con nome

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

Ottenere scheda foglio attivo

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

Inserisci colonna

```
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
```

Inserisci riga

```
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
```

Valore della cella

```
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);
```

Copia celle

```
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));
```

Formula

```
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)");
```

Copia un valore da un foglio al foglio corrente

Immagina di avere un foglio di calcolo Google separato e abbiamo bisogno di portare il valore della cella B2 alla cella D5 sul tuo foglio corrente.

```
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
}
```

Puoi trovare l'ID del foglio di lavoro dal tuo URL.

Ottieni l'ultima riga in una singola colonna

```
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;
    }
  }
}
```

Inserimento di array come righe

Inserire una riga nella parte inferiore di un foglio di calcolo è semplice:

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

Nota questo aggiungerà la riga dopo l'ultima riga *non vuota* .

Inserire una riga da qualche parte nel mezzo è un po 'più di lavoro:

```
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!
```

Un sacco di questo codice inutile può essere estratto in una funzione di aiuto:

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

Il che riduce il nostro esempio a solo:

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

Leggi Servizio di foglio di calcolo online: https://riptutorial.com/it/google-apps-script/topic/2688/servizio-di-foglio-di-calcolo

Capitolo 14: Servizio DriveApp

Osservazioni

I tipi Mime di Google non possono essere utilizzati per il terzo parametro dei Tipi Mime. L'utilizzo di un tipo Mime di Google genera un errore che indica:

Non è possibile utilizzare "DriveApp.createFile ()" per creare tipi MIME di Google. Si prega di utilizzare Advanced Drive Service

MimeType.GOOGLE_APPS_SCRIPT

MimeType.GOOGLE_DOCS

MimeType.GOOGLE_DRAWINGS

MimeType.GOOGLE_FORMS

MimeType.GOOGLE_SHEETS

MimeType.GOOGLE_SLIDES

Examples

Crea una nuova cartella nell'unità radice di 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
};
```

Crea un nuovo file in Google Drive di un certo tipo 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
};
```

Crea un nuovo file di testo nella cartella Google root 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
};
```

Crea un nuovo file in Google Drive da 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
};
```

Ottieni tutte le cartelle - inserisci le cartelle in un token di continuazione - quindi recupera dal token

```
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
};
```

Ottieni tutti i file - inseriscili in un token di continuazione, quindi recuperali

```
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);
};
```

Leggi Servizio DriveApp online: https://riptutorial.com/it/google-apps-script/topic/6395/servizio-driveapp

Capitolo 15: Servizio DriveApp: file per tipo e stringa di ricerca

Parametri

Nome del parametro	Utilizzare per
stringa di ricerca	la stringa che si trova nel nome del file

Examples

Ottieni file per tipo di file con stringa corrispondente nel nome del file

Ottieni tutti i moduli Google con la parola "Senza titolo" nel nome del file.

```
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;
};
```

Leggi Servizio DriveApp: file per tipo e stringa di ricerca online: https://riptutorial.com/it/google-apps-script/topic/4049/servizio-driveapp--file-per-tipo-e-stringa-di-ricerca

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con google-apps-script	Albert Portnoy, Community, Douglas Gaskell, iJay, MShoaib91, Rubén, Saloni Vithalani, Shyam Kansagra, Spencer Easton, sudo bangbang, Supertopoz
2	Aggiungi foglio di calcolo	Bishal, iJay, nibarius
3	App Script Web App	Douglas Gaskell
4	Client chiama a Google apps-script	Supertopoz
5	Crea una funzione personalizzata per Fogli Google	Francky_V, Joshua Dawson, Pierre-Marie Richard, Rubén
6	DriveApp	Brian, Kos, nibarius, Sandy Good, Wolfgang
7	DriveApp - getFileById (id)	Sandy Good
8	Firebase e AppScript: Introduzione	Joseba, Vishal Vishwakarma
9	Foglio attivo di SpreadsheetApp	iJay
10	GmailApp	nibarius
11	Google invia MailApp	Bhupendra Piprava, Brian, Jordan Rhea, Kos, nibarius, Saloni Vithalani
12	Script di Google Web App per il download automatico da Google Drive	Walter
13	Servizio di foglio di calcolo	cdrini, iJay, nibarius, Sandy Good, sudo bangbang
14	Servizio DriveApp	Sandy Good

15	Servizio DriveApp: file per tipo e stringa di ricerca	nibarius, Sandy Good