PHP

Google Sheets – Read & Write Data using Apps Script and PHP

SAJID · UPDATED ON APRIL 4, 2023 · LEAVE A COMMENT

Are you looking to read and write data on Google Sheets using PHP? Occasionally, on your web application, you need to build a flow where you are required to programmatically send data to the Google Sheets and read it. There are 2 possible ways to achieve this goal.

- Using the Google OAuth and google/apiclient PHP library.
- Using a Google Apps Script.

I have tried both options and found Google Apps Script much easier.

In Google OAuth, you have to register your app, build the OAuth flow, store the access token in the database, refresh it in the background and perform the Google API operations. This build requires resources like a database, Google's PHP library, HybridAuth library(to manage OAuth) and quite complicated PHP code.

On the other hand, you will need to add a very little Apps Script code(which is mostly written in JavaScript), get a Web APP URL and hit these URLs for read and write operations. You won't require a database and PHP libraries to accomplish the task. You just have to write GET and POST requests using PHP cURL and you're done.

That being said, let's take a look at how to read and write data on Google Sheets using PHP and Apps Script. We'll create 2 Web App URLs. One is for the POST request to write data on a spreadsheet and another is a GET request to read data from a spreadsheet.

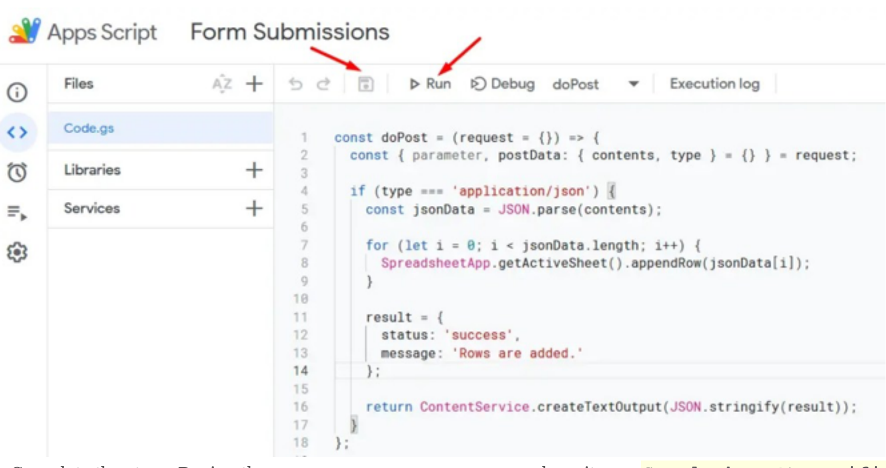
Create Web APP URL for POST Request

Head over to Google Drive and create new Google Sheets. You may add your headings on the first row. Our code will append new entries starting from the next row.

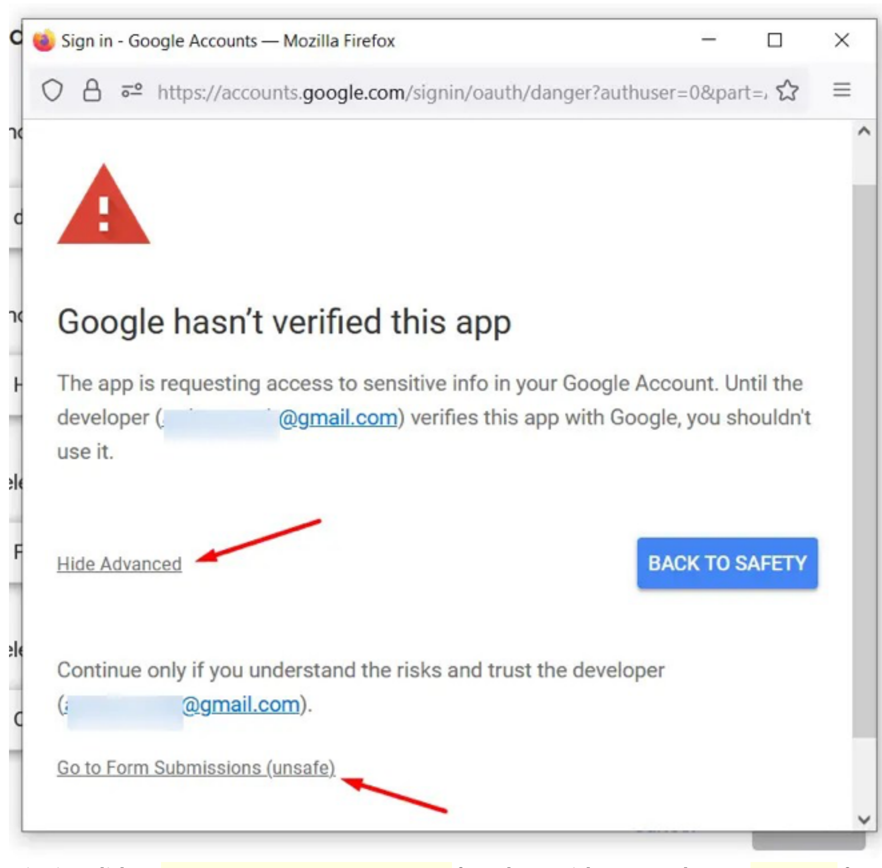
Click on `Extensions -> Apps Script` which will open a new page that has a code editor. Inside the editor, we'll write a code for both GET and POST requests. Let's first handle the POST request.

```
const doPost = (request = {}) => {
  const { parameter, postData: { contents, type } = {} } = request;

  if (type === 'application/json') {
    const jsonData = JSON.parse(contents);

    for (let i = 0; i < jsonData.length; i++) {
    SpreadsheetApp.getActiveSheet().appendRow(jsonData[i]);
    }

    result = {
    status: 'success',
    message: 'Rows are added.'
    };

    return ContentService.createTextOutput(JSON.stringify(result));
  }
};
```

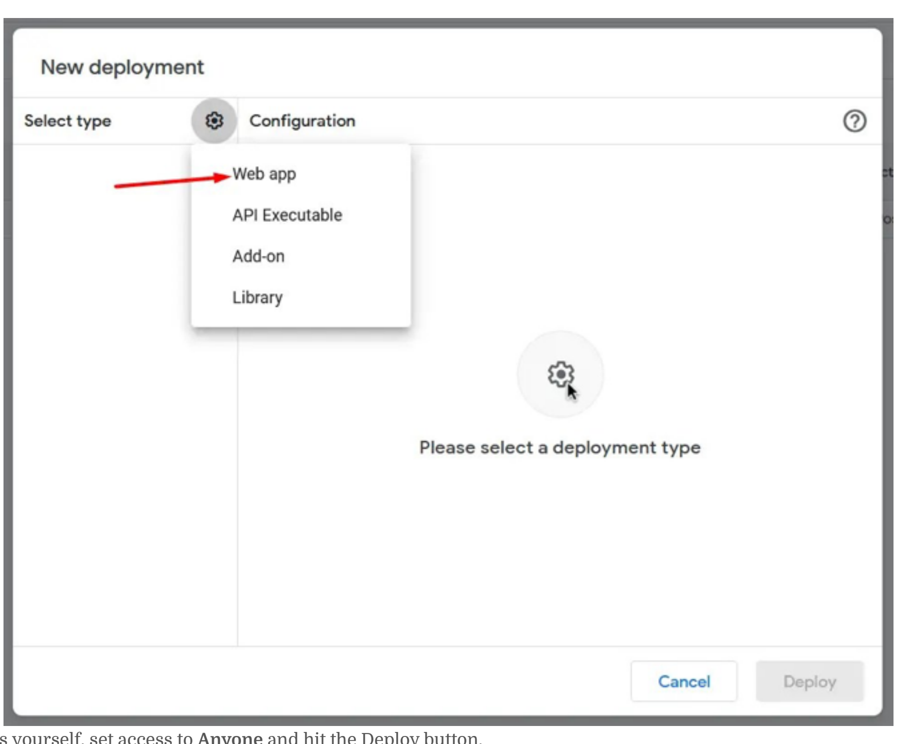
This code receives the payload and writes it to the connected Google Sheets. Our job is to send the payload from the PHP script. And to send the payload we have to generate a Web App URL. For this, save the Apps Script code and click on **Run**.



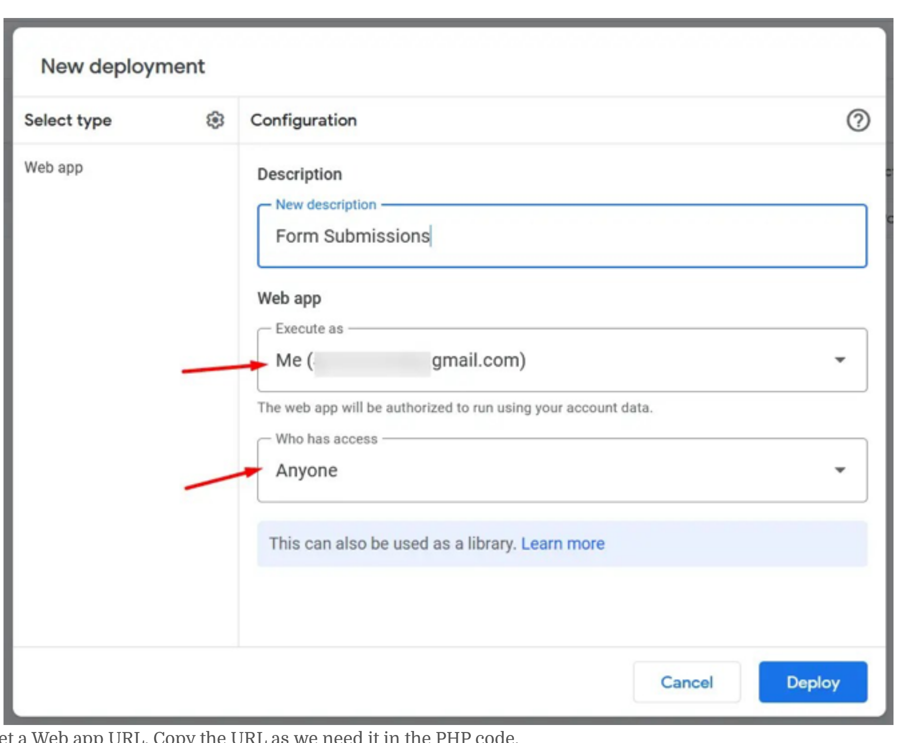
It'll prompt for Authorization. Complete the steps. During the process, you may see a screen where it says `Google hasn't verified this app`. Don't worry about this screen as you're the one who asks for permission. Click on **Advanced** and proceed with it.



Once you complete the authorization, click on `Deploy -> New deployment` from the top right corner. Choose a `Web app` from the settings icon.



Next, execute the web app as yourself, set access to **Anyone** and hit the Deploy button.



In the next window, you'll get a Web app URL. Copy the URL as we need it in the PHP code.

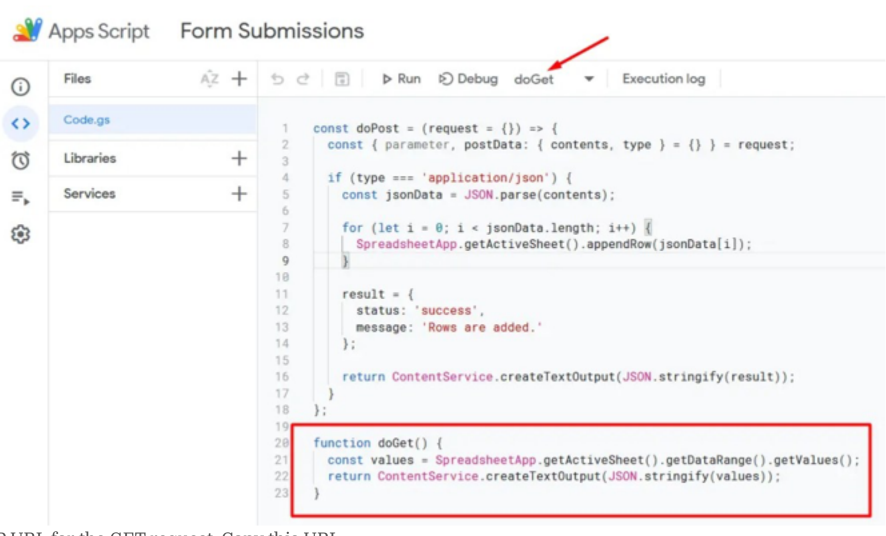
Create Web APP URL for GET Request

Similar to POST requests, you need to generate a Web APP URL for a GET request. In this case, you don't need to perform authorization steps as it has been done already.

Write the below Apps Script code to the same editor after the `doPost` method.

```
function doGet() {
  const values = SpreadsheetApp.getActiveSheet().getDataRange().getValues();
  return ContentService.createTextOutput(JSON.stringify(values));
}
```

After this, set `doGet` at the function dropdown and perform the Deploy process as mentioned in the previous step.



You'll get a different Web APP URL for the GET request. Copy this URL.

Now we have 2 Web APP URLs. One is to write data on Google Sheets and the second one is to read data from Google Sheets.

In the next section, let's see how to handle these GET and POST requests using PHP.

Read and Write Data on Google Sheets Using PHP

To post data on the Google Sheets, we need to JSON-encode the input and send it to the Web APP URL. The Apps Script then automatically appends the entries to the Google Sheets. The below code is written using PHP cURL to send a POST request.

```php
<?php
$url = "WEB_APP_URL_FOR_POST_REQUEST";

$data = array(
    array(
        'John Doe',
        'john.doe@test.com',
        'Web Development',
        'Want to develop a new website.',
    ),
    array(
        'Sam Doe',
        'sam.doe@test.com',
        'Game Development',
        'Want to develop a new video game.',
    ),
);
$payload = json_encode($data);

$ch = curl_init($url);
curl_setopt($ch, CURLOPT_POSTFIELDS, $payload);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true); // follow redirects response
curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type:application/json'));
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$result = curl_exec($ch);
curl_close($ch);
$response = json_decode($result);
if ('success' == $response->status) {
    echo "Form is submitted successfully.";
} else {
    echo "Something went wrong. Try again later.";
}
```

Here, I am passing 2 array elements at a time. Needless to say, you can send as much data as you want.

Go ahead and run this PHP script. You should see your records added to the Google Sheets.

Similarly, using a PHP cURL you can send a GET request and receive data written on Google Sheets.

```php
<?php
$url = "WEB_APP_URL_FOR_GET_REQUEST";

$ch = curl_init($url);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true); // follow redirects response
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$result = curl_exec($ch);
curl_close($ch);
$response = json_decode($result);
print_r($response);
```

If you liked this article, then please subscribe to our YouTube Channel for video tutorials.