Google Cloud Firestore Document CRUD with PHP

There were no examples online, so here you go...

Jason Byrne Follow
Oct 16, 2017 · 5 min read

There is no love for PHP with Google's newest addition Firestore. And it is unlikely there is going to be since Firebase never got an official PHP SDK. In that case Firebase has been around a long time so there are enough open source libraries created by other users to choose from.

But Firestore is so new that I found none. So being that we will soon have the need to write to Firestore from one of our PHP applications... well I decided to create the bones of one using the REST API documentation. There were no examples online outside of that!

First thing that you need to do is get authenticated. The easiest way is to create an API Key, which you can do by logging into this section of your Google Cloud Console: https://console.cloud.google.com/apis/credentials

Now with your API key and project ID in hand, download this rpt of the basic library I fleshed out and save it as *firestore.php*. Now let me get out of the way... this is not a mature library and it is not especially well written, full featured, or abstracted as it should be. I just roughed out this concept in a couple of hours, so get off my back!

```
<?php

include('firestore.php');

use FSPFirestore\FirestoreApiClient;
use FSPFirestore\FirestoreDocument;

$firestore = new FirestoreApiClient(
'YOUR-PROJECT-ID', 'YOUR-API-KEY'
);

$document = new FirestoreDocument();
$document->setString('name', 'Jason');

// Create or update document people/me
$firestore->updateDocument('people', 'me', $document);

// Patch the document people/me
$firestore->patchDocument('people', 'me');

// Remove the document people/me
$firestore->deleteDocument('people', 'me');

// Add this new document in people collection.
// but let Firestore give it the document id
$firestore->addDocument('people', $document);

// Create this document, fail if it already exists
$firestore->updateDocument('people', 'me', $document, false);

// Update this document, fail if it does not exist
$firestore->updateDocument('people', 'me', $document, true);
```

Well you can dig into the source and find out most things, but I'll go over a few interesting high points.

The API root is this: https://firestore.googleapis.com/v1beta1/

After that we append a path that tells Firestore what project and database we are addressing. Apparently at some point each Firestore project will have the capability to have more than one database. For now it is always *(default)* and so the first part of our path is like this:

projects/{project_id}/databases/(default)/

After that we get into actually addressing our resource. In this tutorial, I am only dealing with documents and not collections or indexes or anything. So the next part is only going to be *documents* like so:

projects/{project_id}/databases/(default)/documents/

After that you add your collection name. In my case the collection name is *people* so our path is extended to be:

projects/{project_id}/databases/(default)/documents/people/

Creating a Document

If you adding a document without defining the document id then you will just use a POST to that path. And you will post a JSON formatted payload. This part tripped me up, because at first I just wanted to post something like this:

```
{
  "firstName": "Jason",
  "lastName": "Byrne"
}
```

Whereas that would have probably worked with Firebase Realtime Database, that absolutely did not work here. Reason? The fields have more specific value types, so you have to actually define that like so:

```
{
  "fields": {
    "firstName": {
      "stringValue": "Jason"
    },
    "lastName": {
      "stringValue": "Byrne"
    },
    "age": {
      "integerValue": 36
    }
  }
}
```

To be complete all of the possible "value options are:

- stringValue
- doubleValue
- integerValue
- booleanValue
- arrayValue
- bytesValue
- geoPointValue
- mapValue
- nullValue
- referenceValue
- timestampValue

We will continue using that same document format throughout, no matter which method.

Updating a Document

So with an update (and all the rest of the methods below) we are actually addressing a specific document with a certain name. This update can either be creating a new document (an insert) or updating an existing one.

Here we are using a PATCH which means that if it already exists, we are overlaying these values. In other words, we're not replacing the object (delete and then write again) we are updating it in place. And if we leave off a field, it will not remove it.

So our path just continues where we left off but adds the document id, like this:

projects/{project_id}/databases/(default)/documents/people/jason.byrne

We will simply do a PATCH with the same document format as in our POST above, and again this will be like an update.

Firestore does provide a way to make a strict insert (meaning it would fail if it already existed) by adding

```
?currentDocument.exists=true
```

to the query string. Or if you wanted to make it a strict update (meaning it would fail if it did not exist yet) by adding, of course, the opposite,

```
?currentDocument.exists=false
```

to the query string. Leaving this property off will mean that it will create OR update the document with that name. Also known as an *upsert*.

What you will receive back will look something like this:

```
{
  "name":
"projects/{project}/databases/(default)/documents/{collection}/{docu
ment}",
  "fields": {
    "name": {
      "stringValue": "Jason"
    }
  },
  "createTime": "2017-10-19T00:20:19.285570Z",
  "updateTime": "2017-10-19T00:20:19.285570Z"
}
```

Deleting the Document

Same as a GET or a PATCH, but it would use— obviously — the HTTP DELETE method instead. There was no gotchas that I found with it.

Replacing the Document (PUT or set)

So I don't have a good answer for this one. Logic would say that you could just do a PUT instead of a PATCH and it would replace (set) the document, rather than doing the overlay (update). However, for some reason this method doesn't seem to be supported.

I believe if you were to do a set, you'd have to do it as a batch or something like that. But to be honest, I found it annoying that they didn't support the PUT. And further, I am just too impatient to figure it out right now. And the PATCH should suit my needs fine... so you figure it out if you need it!

Here is a starting point: https://cloud.google.com/firestore/docs/reference/rest/v1beta1/projects.databases.documents/write

What about the REST?? (get it? ... "rest" like double meaning? #dadjoke)

This is just the basics. I didn't even get into querying lists of data. Maybe they'll be a continuation or maybe not. This is all I needed for now, and I just hope that it helps someone else get a place to start from!

Thanks's alll folks!

Firestore Firebase Google Cloud Platform PHP Rest Api