Google Cloud Firestore Document CRUD with PHP There were no examples online, so here you go...

Jason Byrne Follow





ition t



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 <u>REST API documentation</u>. There were no examples online outside of that!

that case Firebase has been around a long time so there are enough open

source libraries created by other users to choose from.

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

Credentials OAuth consent screen Domain verification Credentials

Delete

Requests user consent so your app can access the user's data

API key Identifies your project using a simple API key to check quota and access

OAuth client ID

<?php

include ('firestore.php');

use PHPFireStore\FireStoreApiClient; use PHPFireStore\FireStoreDocument;

\$firestore = new FireStoreApiClient(

Create credentials 🔻

```
Service account key
      Enables server-to-server, app-level authentication using robot accounts
     Help me choose
     Asks a few questions to help you decide which type of credential to use
         OUUGIC OCI VICE
Now with your API key and project ID in hand, download this gist 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!
```

'YOUR-PROJECT-ID', 'YOUR-API-KEY'); \$document = new FireStoreDocument(); \$document->setString('name', 'Jason'); // Create or update document people/me \$firestore->updateDocument('people', 'me', \$document); // Fetch the document people/me \$firestore->getDocument('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 aways (default) and so the first part of our path is like this:

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

After that you add your collection name. In my case the collection name is

projects/{project_id}/databases/(default)/documents/people/ **Creating a Document**

Whereas that would have probably worked with Firebase Realtime

specific value types, so you have to actually define that like so:

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:

"fields": { "firstName": { "stringValue": "Jason" "lastName": { "stringValue": "Byrne"

Database, that absolutely did not work here. Reason? The fields have more

```
    integerValue

    booleanValue

    arrayValue

    bytesValue

    geoPointValue

    mapValue

    nullValue

    referenceValue
```

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

So with an update (and all the rest of the methods below) we are actually

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

addressing a specific document with a certain name. This update can either

to the query string. Leaving this property off will mean that it will create OR

This is so straight forward that you don't need any much explanation of it.

You just use the same path as you would with your update above, just do an

update the document with that name. Also known as an upsert.

What you will receive back will look something like this:

Firestore does provide a way to make it a strict insert (meaning it would fail

"stringValue": "Jason" "createTime": "2017-10-19T00:20:19.268570Z", "updateTime": "2017-10-19T00:20:19.268570Z"

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)

265 Q 11

Google Cloud Platform

Founder of MileSplit.

FloSports Engineering Experience, ideas, half-baked thoughts, and ramblings from the coders, testers, devops, hackers, and geeks at FloSports. http://www.flosports.tv

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

people so our path is extended to be:

the next part is only going to be documents like so:

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

"firstName": "Jason", "lastName": "Byrne"

"integerValue": 36

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

"age": {

stringValue

doubleValue

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

timestampValue

Updating a Document

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.

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.

if it already existed) by adding

?currentDocument.exists=true

Getting the Document

HTTP GET method instead.

"name":

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

"projects/{project}/databases/(default)/documents/{collection}/{docou "fields": { "name": {

Same as a GET or a PATCH, but it would use—obviously—the HTTP

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,

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!

DELETE method instead. There was no gotchas that I found with it.

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

Thatttt's allll folks!

Firebase

Deleting the Document

Replacing the Document (PUT or set)

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!

Rest Api

WRITTEN BY Jason Byrne Entrepreneur, developer, historian, journalist, Christian, family

man, and track & field fan. VP, Engineering @ FloSports.

Giving a spin to Cloud Uploading Image to the Datastream, The new Firebase Storage using serverless CDC offering Cloud functions from on Google cloud NodeJS Environment. Nikhil Koduri in Searce Jane Uchechukwu in The Startup

How to upload data to Firebase Firestore Cloud Database

More From Medium

Lastinger Family were

Jason Byrne in Florida History

first to settle pioneer

Lake Placid

Devesu

Learn more.

• Medium





inbox. Explore

Exporting Firestore

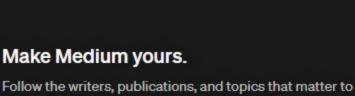
Collection CSV into

Demand, the easy way

Cloud Storage on

Lim Shang Yi in Google

Developers Experts



About

Full Scalable Angular

application based on

Migrate your Firebase

Doug Stevenson in Firebase

Cloud Functions to

Node.js 10

Developers

AWS

Alban

Infinite Scroll

Follow

Follow

Help

Legal

nade

Medium is an open platform where 170 million readers

and undiscovered voices alike dive into the heart of any

come to find insightful and dynamic thinking. Here, expert

Infinity Scroll in Flutter

Anggardha Febriano

using Bloc and Firestore