CouchDB for PHP developers – CRUD

By: Branko Ajzele, Aug 27, 2013 Dev Talk
https://inchoo.net/dev-talk/couchdb-for-php-developers-crud/

Apache CouchDB™ commonly referred to as just CouchDB is an open source NoSQL database that uses JSON for storing data in form of documents, JavaScript as its query language using MapReduce queries, and regular HTTP for an API.

There is an excellent, free book called CouchDB The Definitive Guide available online at http://guide.couchdb.org. If you haven't read it, I strongly advise doing so.

Working with JavaScript Object Notation (aka JSON) in PHP is a breeze. The best thing about JSON in PHP is that there is no required installation of external libraries needed, as JSON support is part of the PHP core:

- json_decode – Decodes a JSON string
- json_encode – Returns the JSON representation of a value

When it comes to API communication through HTTP, you are also pretty well covered here. PHP supports libcurl, a library that allows you to connect and communicate to many different types of servers with many different types of protocols. Among other protocols, libcurl supports supports http and https. cURL functions is probably more familiar term for this. Although this is such a frequently used library, some servers might not have it installed, so be sure to check if cURL is installed (for example, function_exists('curl_version')).

The Futon

Another great thing about CouchDB is that it comes with a native web-based interface built into it, sort of like phpMyAdmin for MySQL. This interface is called Futon. It provides a basic interface to the majority of the functionality, including the ability to create, update, delete and view documents and views. Reason why we are bringing this up, is to that you can check the results of all of the API calls we have run so far. For more information on Futon itself, check out the official documentation page http://docs.couchdb.org/en/latest/intro.html#using-futon. You should be able to access Futon via the following URL http://127.0.0.1:5984/_utils/.

So now that we have a support JSON handling, HTTP communication and a interface to check our results, we

can move on with some practical examples.
Note: In most of the following examples, I am assuming that you used the Futon to set the username/password on a database. If you haven't, you can comment out the line of code with CURLOPT_USERPWD.
Now, lets jump on to some practical examples.

Check if CouchDB is running

You can check if CouchDB is running just by executing HTTP GET request at http://127.0.0.1:5984 URL. If all is OK, you should be able to see a "Welcome" message as a part of larger JSON object.

```
<?php

$ch = curl_init();

curl_setopt($ch, CURLOPT_URL, 'http://
127.0.0.1:5984/');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    'Content-type: application/json',
    'Accept: */*'
));

$response = curl_exec($ch);

curl_close($ch);

/* JSON string
{
  "couchdb": "Welcome",
  "uuid": "fd53dde57b4f2ead04267c54139358a7",
  "version": "1.3.1",
  "vendor": {
    "version": "1.3.1",
    "name": "The Apache Software Foundation"
  }
```

```
}
*/
```

Get a list of databases

Try not to get confused with terminology “CouchDB” vs “databases”. CouchDB is a database management system (DMS), which means it can hold multiple databases. A database is a bucket that holds “related data”, such as products.

```php
<?php

$ch = curl_init();

curl_setopt($ch, CURLOPT_URL, 'http://127.0.0.1:5984/
_all_dbs');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    'Content-type: application/json',
    'Accept: */*'
));

$response = curl_exec($ch);

curl_close($ch);

/*
string(36) "["_replicator","_users","products"] "
*/
```

Create a database

In this example we will create an empty database called customers. For this, we need to use HTTP PUT request.

```php
<?php

$table = ‘customers’;
$ch = curl_init();

curl_setopt($ch, CURLOPT_URL, 'http://127.0.0.1:5984/'.
```

```
$table);
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'PUT');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_USERPWD,
'myDBusername:myDBpass');
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    'Content-type: application/json',
    'Accept: */*'
));

$response = curl_exec($ch);

curl_close($ch);

/*
string(12) "{"ok":true} "
*/
```

Get an UUID from CouchDB

If you find it impractical to generate your own UUID's, you can ask CouchDB to give you one. If you need more than one UUID, you can pass in the ?count=5 HTTP parameter to request 5 UUIDs, or any other number you need.

```
<?php

$ch = curl_init();

curl_setopt($ch, CURLOPT_URL, 'http://127.0.0.1:5984/
_uuids');
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'GET');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    'Content-type: application/json',
    'Accept: */*'
));

curl_setopt($ch, CURLOPT_USERPWD,
'myDBusername:myDBpass');
```

```php
$response = curl_exec($ch);
$_response = json_decode($response, true);

$UUID = $_response['uuids'];

curl_close($ch);

/*
$UUID: string(32) "c2449e8c6cb8cb938ee507e281003348"
*/
```

Create a document

To create new document you can either use a POST operation or a PUT operation.
In this example we are creating a document (an entry, a row, a record) within the customers database.
Each document in CouchDB has an ID. This ID is unique per database. You are free to choose any string to be the ID, although CouchDB recommends a UUID (or GUID). In the example above I showed you how to fetch the UUID from CouchDB itself. Since ID is a required parameter that needs to be passed with create a document request, we can either:
request it from CouchDB
use some other unique string for it.
For our customers table, we will use username field for ID as shown in the example below. Please note that this is not the best decision, as it is recommended to use the UUID for ID. However, for the sake of simplicity and easier overview we will use username.

```php
<?php

$ch = curl_init();
```

```php
$customer = array(
    'firstname' => 'Branko',
    'lastname' => 'Ajzele',
    'username' => 'ajzele',
    'email' => 'branko.ajzele@example.com',
    'pass' => md5('myPass123')
);

$payload = json_encode($customer);

curl_setopt($ch, CURLOPT_URL, 'http://127.0.0.1:5984/
customers/'.$customer['username']);
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'PUT'); /* or
PUT */
curl_setopt($ch, CURLOPT_POSTFIELDS, $payload);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    'Content-type: application/json',
    'Accept: */*'
));

curl_setopt($ch, CURLOPT_USERPWD,
'myDBusername:myDBpass');

$response = curl_exec($ch);

curl_close($ch);

/*
string(69) "{"ok":true,"id":"ajzele","rev":"1-
c3fde3a56fe3c3490448a8e34166b4ec"} "
*/
```

Get a document

To retrieve a document, simply perform a GET operation at the document's URL like shown below. Here we are using $documentID = 'ajzele'; to fetch the document we just created in previous example.

```php
<?php

$ch = curl_init();

$documentID = 'ajzele';

curl_setopt($ch, CURLOPT_URL, 'http://127.0.0.1:5984/
customers/'.$documentID);
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'GET');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    'Content-type: application/json',
    'Accept: */*'
));

curl_setopt($ch, CURLOPT_USERPWD,
'myDBusername:myDBpass');

$response = curl_exec($ch);

curl_close($ch);

/*
string(200) "{"_id":"ajzele","_rev":"1-
c3fde3a56fe3c3490448a8e34166b4ec","firstname":"Branko",
"lastname":"Ajzele","username":"ajzele","email":"branko
.ajzele@example.com","pass":"433484b5317340f5c28e085bff
fc78be"} "
*/
```

Update existing document

Note the _rev value 1-c3fde3a56fe3c3490448a8e34166b4ec in Create a document example. Prefix 1 means first revision. If you try to execute the same create a document request multiple times, you would get Document update conflict. error message. This is because CouchDB sees your request as update. If you want to update or delete a document,

CouchDB expects you to include the _rev field of the revision you wish to change. When CouchDB accepts the change, it will generate a new revision number.
In order to update our previously create document, we need to pass it the _rev number we got from CouchDB when we created the document. Below is an example of such request, note how it is almost identical.

```
<?php

$ch = curl_init();

$customer = array(
    'firstname' => 'Branko_2',
    'lastname' => 'Ajzele_2',
    'username' => 'ajzele',
    'email' => 'branko.ajzele@example.com',
    'pass' => md5('myPass123')
);

$customer['_rev'] = '1-
c3fde3a56fe3c3490448a8e34166b4ec';

$payload = json_encode($customer);

curl_setopt($ch, CURLOPT_URL, 'http://127.0.0.1:5984/
customers/'.$customer['username']);
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'PUT'); /* or
PUT */
curl_setopt($ch, CURLOPT_POSTFIELDS, $payload);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    'Content-type: application/json',
    'Accept: */*'
));

curl_setopt($ch, CURLOPT_USERPWD,
'myDBusername:myDBpass');
```

```
$response = curl_exec($ch);

curl_close($ch);

/*
string(69)
"{"ok":true,"id":"ajzele","rev":"2-6a7e68954964a35d9415
eb10142d17fc"} "
*/
```

Creating a document with attachment

CouchDB documents can have attachments. They can be any data (pdf, image, music, video...). It is important to know that attachments are added only to an existing documents. Process of adding an attachment is considered a document update. Since each document update requires a revision number, so does the process of adding attachment.

```
<?php

$ch = curl_init();

$database = 'customers';
$documentID = 'ajzele';
$revision = '2-6a7e68954964a35d9415eb10142d17fc';

$attachment = 'inkscape-0.48.4-1-win32.exe';
$repository = 'C:/Users/branko/Downloads/';

$finfo = finfo_open(FILEINFO_MIME_TYPE);
$contentType = finfo_file($finfo, $repository.
$attachment);

$payload = file_get_contents($repository.$attachment);

curl_setopt($ch, CURLOPT_URL, sprintf('http://
127.0.0.1:5984/%s/%s/%s?rev=%s', $database,
$documentID, $attachment, $revision));
```

```
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'PUT');
//curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, $payload);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    'Content-type: '.$contentType,
    'Accept: */*'
));

curl_setopt($ch, CURLOPT_USERPWD,
'myDBusername:myDBpass');

$response = curl_exec($ch);

curl_close($ch);

/*
string(69)
"{"ok":true,"id":"ajzele","rev":"3-4b53bf95181e2461cc4b
417507cf1e45"} "
*/
```

Attachments are stored under the root _attachments key of a JSON document, in a form like shown below.

```
"_attachments": {
   "inkscape-0.48.4-1-win32.exe": {
       "content_type": "application/x-dosexec",
       "revpos": 5,
       "digest": "md5-4lZleB3ePMxCaF7QraQEQg==",
       "length": 34702513,
       "stub": true
   },
   "Magento_Community_1-7_User_Guide.zip": {
       "content_type": "application/zip",
       "revpos": 3,
       "digest": "md5-j9E1m4ejrc7LtlRuKgq8cA==",
       "length": 9034796,
       "stub": true
   }
}
```

Delete a document

To delete a document you need to perform a HTTP DELETE operation at the document's location, passing the rev parameter with the document's current revision.

```php
<?php

$ch = curl_init();

$database = 'customers';
$documentID = 'ajzele';
$revision = '7-6f74f3f827fa594ef7bf3293490e5d79';

curl_setopt($ch, CURLOPT_URL, sprintf('http://
127.0.0.1:5984/%s/%s?rev=%s', $database, $documentID,
$revision));
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'DELETE');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    'Content-type: application/json',
    'Accept: */*'
));

curl_setopt($ch, CURLOPT_USERPWD,
'myDBusername:myDBpass');

$response = curl_exec($ch);

curl_close($ch);
/*
string(69)
"{"ok":true,"id":"ajzele","rev":"8-80af68b3814813ebc57b
b70e73b8524a"} "
*/
```

Interesting enough that the DELETE action, besides just deleting the document from database, also returns the increased revision number.

Summary

In this article we covered several basic but essential CRUD use cases, based on which you can start building your own application. Even doe there are already several PHP libraries out there available for working with CouchDB, we focused on doing the same with basic cURL as this is something every PHP developer should know how to do. The next follow up article **CouchDB for PHP developers – Searching/Querying** will cover CouchDB Map/Reduce functions that we will use for Searching/Querying the database.

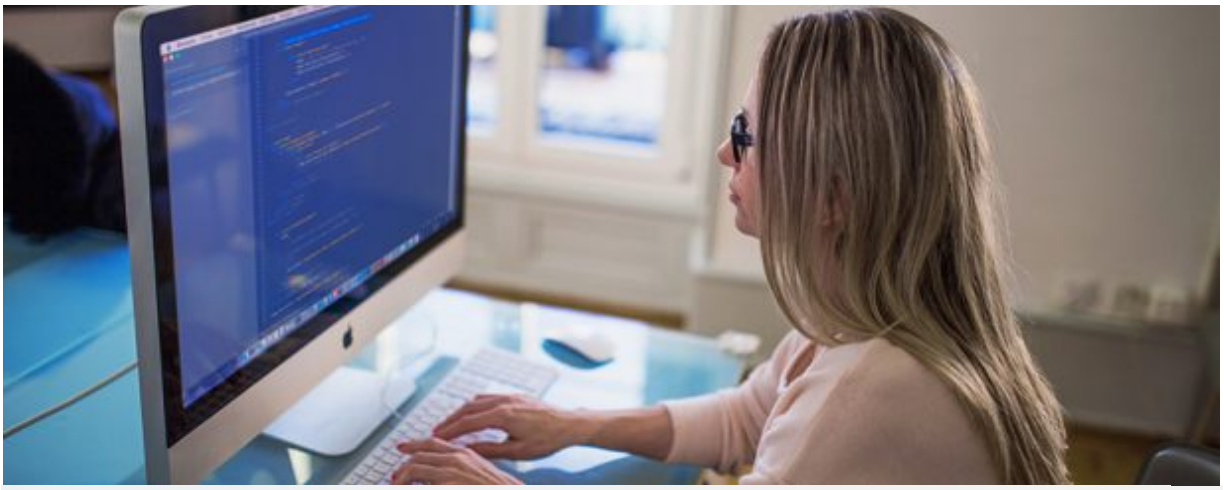
Published in: Dev Talk

LEAVE A COMMENT

YOU MADE IT ALL THE WAY DOWN HERE SO YOU MUST HAVE ENJOYED THIS POST! YOU MAY ALSO LIKE:

Josip Kovacevic, Sep 30, 2019 | 3

Declarative Schema feature in Magento 2



Matej Maricic, Aug 25, 2019 | 10

Magento 2 logging

Deni Pesic, Jun 06, 2018 | 2

GDPR compliant Magento 2 database dump

16 comments

Subhra Jyoti Lahiri says:
Sep 23, 2019 at 7:31

Do we need to write all the field parameters in order to update the document. I mean to say, to update the last name only, we need to send the full Object along with the changes to reflect it??

Reply

bilal says:
Sep 20, 2018 at 22:37

not running even any of the function

Reply

yhosymar says:
Sep 16, 2018 at 23:56

can you help me, i want create bulk document whith this example, but i cant

Reply

Adrian says:
Mar 14, 2016 at 10:15

class ....

```
<?php

class Wbb_CouchDB
{

    /**
     * The Server API Url .
     *
     * @var string
     */
    public $server_api_url;

    /**
```

```
         * The Server Database Username
         *
         * @var bool
         */
        private $_my_db_username;

        /**
         * The Server Database Password
         *
         * @var bool
         */
        private $_my_db_password;

        /**
         * Curl Connection instance
         *
         * @var
         */
        private $_curl_initialization;

        /**
         * Wbb_CouchDB constructor.
         *
         * @param string $server_api_url
         * @param bool   $my_db_username
         * @param bool   $my_db_password
         */
        public function __construct ( $server_api_url
    = '' , $my_db_username = FALSE , $my_db_password =
    FALSE )
        {

            // The Server API Url .
            $this->server_api_url = $server_api_url;

            // The Server Database Username
            $this->_my_db_username = $my_db_username;

            // The Server Database Password
            $this->_my_db_password = $my_db_password;
```

```
    }

    /**
     * Initialize Curl Connection .
     */
    public function InitConnection ()
    {

        $this->_curl_initialization = curl_init
();

    }

    /**
     * Close Curl Connection
     */
    public function CloseConnection ()
    {
        curl_close ( $this-
>_curl_initialization );
    }

    /**
     * Check if CouchDB is running
     *
     * You can check if CouchDB is running just by
executing HTTP GET request at http://127.0.0.1:5984
URL. If all is
     * OK, you should be able to see a “Welcome”
message as a part of larger JSON object.
     *
     * @return mixed
     */
    public function isRunning ()
    {

        curl_setopt ( $this-
>_curl_initialization , CURLOPT_URL , $this-
>server_api_url );
        curl_setopt ( $this-
>_curl_initialization , CURLOPT_RETURNTRANSFER ,
```

```
TRUE );
        curl_setopt (
            $this->_curl_initialization ,
CURLOPT_HTTPHEADER , array (

'Content-type: application/json' ,

'Accept: */*' ,
                                        )
        );

        $response = curl_exec ( $this-
>_curl_initialization );

        return $response;

    }

    /**
     * Get a list of databases
     *
     * Try not to get confused with terminology
“CouchDB” vs “databases”. CouchDB is a database
management system
     * (DMS), which means it can hold multiple
databases. A database is a bucket that holds
“related data”, such as
     * products.
     *
     */
    public function getAllDbs ()
    {

        curl_setopt ( $this-
>_curl_initialization , CURLOPT_URL , $this-
>server_api_url . '/_all_dbs' );
        curl_setopt ( $this-
>_curl_initialization , CURLOPT_RETURNTRANSFER ,
TRUE );
        curl_setopt (
            $this->_curl_initialization ,
```

```
    CURLOPT_HTTPHEADER , array (

    'Content-type: application/json' ,

    'Accept: */*' ,
                                            )
            );

            $response = curl_exec ( $this-
    >_curl_initialization );

            return $response;
        }

        /**
         * @param $db_name
         *
         * Create a database
         *
         * In this example we will create an empty
    database called customers. For this, we need to use
    HTTP PUT request.
         *
         * @return mixed
         */
        public function createDb ( $db_name )
        {

            curl_setopt ( $this-
    >_curl_initialization , CURLOPT_URL , $this-
    >server_api_url . '/' . $db_name );
            curl_setopt ( $this-
    >_curl_initialization , CURLOPT_CUSTOMREQUEST ,
    'PUT' );
            curl_setopt ( $this-
    >_curl_initialization , CURLOPT_RETURNTRANSFER ,
    TRUE );

            if ( $this->_my_db_password && $this-
    >_my_db_username )
            {
```

```
            curl_setopt ( $this-
>_curl_initialization , CURLOPT_USERPWD , $this-
>_my_db_username . ':' . $this->_my_db_password );
        }

        curl_setopt (
            $this->_curl_initialization ,
CURLOPT_HTTPHEADER , array (

'Content-type: application/json' ,

'Accept: */*' ,
                                        )
        );

        $response = curl_exec ( $this-
>_curl_initialization );

        return $response;

    }

    /**
     * Get an UUID from CouchDB
     *
     * If you find it impractical to generate your
own UUID's, you can ask CouchDB to give you one. If
you need more
     * than one UUID, you can pass in the ?count=5
HTTP parameter to request 5 UUIDs, or any other
number you need.
     */
    public function GetUUID ()
    {

        curl_setopt ( $this-
>_curl_initialization , CURLOPT_URL , $this-
>server_api_url . '/_uuids' );
        curl_setopt ( $this-
>_curl_initialization , CURLOPT_CUSTOMREQUEST ,
'GET' );
```

```
            curl_setopt ( $this-
>_curl_initialization , CURLOPT_RETURNTRANSFER ,
TRUE );
            curl_setopt (
                $this->_curl_initialization ,
CURLOPT_HTTPHEADER , array (

'Content-type: application/json' ,

'Accept: */*' ,
                                                )
            );

            if ( $this->_my_db_password && $this-
>_my_db_username )
            {
                curl_setopt ( $this-
>_curl_initialization , CURLOPT_USERPWD , $this-
>_my_db_username . ':' . $this->_my_db_password );
            }

            $response  = curl_exec ( $this-
>_curl_initialization );
            $_response = json_decode ( $response ,
TRUE );

            $UUID = $_response[ 'uuids' ];

            return $UUID;

        }

        /**
         * Create a document
         *
         * To create new document you can either use a
POST operation or a PUT operation.
         *
         * In this example we are creating a document
(an entry, a row, a record) within the customers
database.
```

```
     *
     * Each document in CouchDB has an ID. This ID
is unique per database. You are free to choose any
string to be the
     * ID, although CouchDB recommends a UUID (or
GUID). In the example above I showed you how to
fetch the UUID from
     * CouchDB itself. Since ID is a required
parameter that needs to be passed with create a
document request, we can
     * either: request it from CouchDB use some
other unique string for it. For our customers
table, we will use
     * username field for ID as shown in the
example below. Please note that this is not the
best decision, as it is
     * recommended to use the UUID for ID.
However, for the sake of simplicity and easier
overview we will use
     * username.
     *
     *
     * @param array $document_data
     * @param       $database
     * @param       $document
     *
     * @return mixed
     */
    public function createDocument
( $document_data = array () , $database ,
$document )
    {

        curl_setopt ( $this-
>_curl_initialization , CURLOPT_URL , $this-
>server_api_url . '/' . $database . '/' .
$document );
        curl_setopt ( $this-
>_curl_initialization , CURLOPT_CUSTOMREQUEST ,
'PUT' ); /* or PUT */
        curl_setopt ( $this-
```

```
>_curl_initialization , CURLOPT_POSTFIELDS ,
json_encode ( $document_data ) );
        curl_setopt ( $this-
>_curl_initialization , CURLOPT_RETURNTRANSFER ,
TRUE );
        curl_setopt (
            $this->_curl_initialization ,
CURLOPT_HTTPHEADER , array (

'Content-type: application/json' ,

'Accept: */*' ,
                                        )
        );

        if ( $this->_my_db_password && $this-
>_my_db_username )
        {
            curl_setopt ( $this-
>_curl_initialization , CURLOPT_USERPWD , $this-
>_my_db_username . ':' . $this->_my_db_password );
        }

        $response = curl_exec ( $this-
>_curl_initialization );

        return $response;
    }

    /**
     * Creating a document with attachment
     *
     * CouchDB documents can have attachments.
They can be any data (pdf, image, music, video…).
It is important to
     * know that attachments are added only to an
existing documents. Process of adding an attachment
is considered a
     * document update. Since each document update
requires a revision number, so does the process of
adding
```

```
         * attachment.
         *
         * @param $database
         * @param $documentID
         * @param $revision
         * @param $attachment
         *
         * @return mixed
         */
        public function createAttachmentDocument
    ( $database , $documentID , $revision ,
    $attachment )
        {

            $finfo       = finfo_open
    ( FILEINFO_MIME_TYPE );
            $contentType = finfo_file ( $finfo ,
    $attachment );
            $payload     = file_get_contents
    ( $attachment );

            curl_setopt ( $this-
    >_curl_initialization , CURLOPT_URL , sprintf
    ( $this->server_api_url . '/%s/%s/%s?rev=%s' ,
    $database , $documentID , $attachment ,
    $revision ) );
            curl_setopt ( $this-
    >_curl_initialization , CURLOPT_CUSTOMREQUEST ,
    'PUT' );
            //curl_setopt( $this-
    >_curl_initialization, CURLOPT_POST, true);
            curl_setopt ( $this-
    >_curl_initialization , CURLOPT_POSTFIELDS ,
    $payload );
            curl_setopt ( $this-
    >_curl_initialization , CURLOPT_RETURNTRANSFER ,
    TRUE );
            curl_setopt (
                $this->_curl_initialization ,
    CURLOPT_HTTPHEADER , array (
```

```
    'Content-type: ' . $contentType ,

    'Accept: */*' ,
                                            )
            );

            if ( $this->_my_db_password && $this-
>_my_db_username )
            {
                curl_setopt ( $this-
>_curl_initialization , CURLOPT_USERPWD , $this-
>_my_db_username . ':' . $this->_my_db_password );
            }

            $response = curl_exec ( $this-
>_curl_initialization );

            return $response;

        }

        /**
         * Get a document
         *
         *
         * To retrieve a document, simply perform a
GET operation at the document's URL like shown
below. Here we are using
         * $documentID = 'ajzele'; to fetch the
document we just created in previous example.
         *
         * @param $database
         * @param $documentID
         *
         * @return mixed
         */
        public function getDocument ( $database ,
$documentID )
        {

            curl_setopt ( $this-
```

```
>_curl_initialization , CURLOPT_URL , $this-
>server_api_url . '/' . $database . '/' .
$documentID );
        curl_setopt ( $this-
>_curl_initialization , CURLOPT_CUSTOMREQUEST ,
'GET' );
        curl_setopt ( $this-
>_curl_initialization , CURLOPT_RETURNTRANSFER ,
TRUE );
        curl_setopt (
            $this->_curl_initialization ,
CURLOPT_HTTPHEADER , array (

'Content-type: application/json' ,

'Accept: */*' ,
                                        )
        );

        if ( $this->_my_db_password && $this-
>_my_db_username )
        {
            curl_setopt ( $this-
>_curl_initialization , CURLOPT_USERPWD , $this-
>_my_db_username . ':' . $this->_my_db_password );
        }

        $response = curl_exec ( $this-
>_curl_initialization );

        return $response;

    }

    /**
     *
     * Update existing document
     *
     * Note the _rev value $document_rev (1-
c3fde3a56fe3c3490448a8e34166b4ec) in Create a
document example. Prefix 1
```

```
     * means first revision. If you try to execute
the same create a document request multiple times,
you would get
     * Document update conflict. error message.
This is because CouchDB sees your request as
update. If you want to
     * update or delete a document, CouchDB
expects you to include the _rev field of the
revision you wish to change.
     * When CouchDB accepts the change, it will
generate a new revision number.
     *
     * In order to update our previously create
document, we need to pass it the _rev number we got
from CouchDB when
     * we created the document. Below is an
example of such request, note how it is almost
identical.
     *
     * @param array $document_data
     * @param       $database
     * @param       $document
     * @param bool  $document_rev
     *
     * @return mixed
     */
    public function updateDocument
( $document_data = array () , $database ,
$document , $document_rev = FALSE )
    {

        if ( $document_rev )
        {

            $document_data[ '_rev' ] =
$document_rev;
        }

        curl_setopt ( $this-
>_curl_initialization , CURLOPT_URL , $this-
>server_api_url . '/' . $database . '/' .
```

```
$document );
        curl_setopt ( $this->_curl_initialization , CURLOPT_CUSTOMREQUEST , 'PUT' ); /* or PUT */
        curl_setopt ( $this->_curl_initialization , CURLOPT_POSTFIELDS , json_encode ( $document_data ) );
        curl_setopt ( $this->_curl_initialization , CURLOPT_RETURNTRANSFER , TRUE );
        curl_setopt (
            $this->_curl_initialization , CURLOPT_HTTPHEADER , array (

'Content-type: application/json' ,

'Accept: */*' ,
                                                )
        );

        if ( $this->_my_db_password && $this->_my_db_username )
        {
            curl_setopt ( $this->_curl_initialization , CURLOPT_USERPWD , $this->_my_db_username . ':' . $this->_my_db_password );
        }

        $response = curl_exec ( $this->_curl_initialization );

        return $response;

    }

    /**
     * Delete a document
     *
     * To delete a document you need to perform a HTTP DELETE operation at the document's location, passing the rev
```

```
     * parameter with the document's current
revision.
     *
     * @param $database
     * @param $documentID
     * @param $revision
     *
     * @return mixed
     */
    public function deleteDocument ( $database ,
$documentID , $revision )
    {

        curl_setopt ( $this-
>_curl_initialization , CURLOPT_URL , sprintf
( $this->_my_db_username . '/%s/%s?rev=%s' ,
$database , $documentID , $revision ) );
        curl_setopt ( $this-
>_curl_initialization , CURLOPT_CUSTOMREQUEST ,
'DELETE' );
        curl_setopt ( $this-
>_curl_initialization , CURLOPT_RETURNTRANSFER ,
TRUE );
        curl_setopt (
            $this->_curl_initialization ,
CURLOPT_HTTPHEADER , array (

'Content-type: application/json' ,

'Accept: */*' ,
                                        )
        );

        if ( $this->_my_db_password && $this-
>_my_db_username )
        {
            curl_setopt ( $this-
>_curl_initialization , CURLOPT_USERPWD , $this-
>_my_db_username . ':' . $this->_my_db_password );
        }
```

```
            $response = curl_exec ( $this-
>_curl_initialization );

            return $response;

        }

}
```

Reply

Markus says:
Jan 06, 2016 at 20:22

Is there a link to the follow up article you mentioned on searching/querying… I was unable to get a hit on that with google or see it in your dev-talk feed… Thanks! Enjoyed this article.

Reply

Marco says:

Sep 01, 2015 at 10:58

Hi 🙂 can you help me? I use XAMPP on my Windows 7 O.S and I've just installed couchdb for windows. How can I create a connection between xampp php and couchdb server?

Reply

php tutorials says:
Jul 06, 2015 at 13:24

Nice blog...Very useful information is providing by ur blog.Very clear and helpful for beginners.

Reply

尤川豪 says:
Apr 16, 2015 at 19:39

Hi!
Thanks for the tutorial!

It’s very helpful for beginners like me :)

Reply

Dimitar Ivanov says:
Jun 22, 2014 at 8:24

HeadCouch – CouchDB PHP client
http://zinoui.com/blog/headcouch-couchdb-php-client
That is a PHP library written by me, for working with CouchDB. It requires PHP5.x and cURL.

Reply

Daniel says:
Mar 28, 2014 at 14:41

Robert… it’s pretty easy to update a document w/ the wrapper I linked. Assuming that you have a “relax” object, it’s:
$document = $db->get_doc(whatever your id is);
$document[key to change’] = “value”;

$db->save($document);

Reply

Jeevs says:
Mar 28, 2014 at 11:21

How can I delete a single field in a document in couchdb with PHP.

Reply

Robert Beran says:
Mar 21, 2014 at 12:20

Hi there ...
if you want to update an document with other values with php ... just do the following ... get rev id firom the document ... and change the value in the json array and put it up ...

```
public function update_data($doc, $jsonkey,
$jsonvalue) {
$resp = $this->couch->send(“GET”, “/somedb/”.$doc);
$arr = json_decode($resp, true);
$arr[$jsonkey] = $jsonvalue;
$update = json_encode($arr);
$this->couch->send(“PUT”, “/somedb/”.$doc, $update);
}
```

and here the class :

```
class CouchSimple {
function CouchSimple($options) {
foreach($options AS $key => $value) {
$this->$key = $value;
}
}
function send($method, $url, $post_data = NULL) {
$s = fsockopen($this->host, $this->port, $errno,
$errstr);
if(!$s) {
echo “$errno: $errstr\n”;
return false;
}
$request = “$method $url HTTP/1.0\r\nHost: $this-
>host\r\n”;
if($post_data) {
$request .= “Content-Length: “.strlen($post_data).”
\r\n\r\n”;
$request .= “$post_data\r\n”;
```

```
}
else {
$request .= “\r\n”;
}
fwrite($s, $request);
$response = “”;
while(!feof($s)) {
$response .= fgets($s);
}
list($this->headers, $this->body) = explode(“\r\n\r\n”,
$response);
return $this->body;
}
}
```

Reply

Daniel says:
Oct 15, 2013 at 16:18

https://github.com/barnjamin/relax/blob/master/relax.php is a simply PHP wrapper for Couchdb. It’s something that I use very frequently and handles a majority of things. There are a couple of errors in the repo that I’ll need to push out there, but nothing that

isn't easy to fix (e.g.: "Etag" vs "ETag"). If you've got and questions about using the wrapper, let me know

```
This is a php class that interacts with couchdb
that doesn't bother with any kind of catching of
errors.
```

```
<?php


  class couch{


      private $options = array(
          'host'=>'localhost',
          'port'=>5984,
          'ip' => '127.0.0.1',
         'timeout' => 2,
         'keep-alive'=> true,
         'http-log'=> false,
          'username'=>null,
          'password'=>null
      );


      public $database;


      public function __construct($username = null,
  $password = null, $db = null , $host = null, $port
  = null, $ip = null) {
          $this->options['host']     =
  (isset($host))?(string)$host:$this-
  >options['host'];
          $this->options['port']     =
  (isset($port))?(int)$port:$this->options['port'];
          $this->options['username'] = $username;
          $this->options['password'] = $password;
```

```
            $this->database = $db;
        }


        /*
        *Creates a database with the name of whatever
is stored in $database
        */
        public function create_db($name=null){
            $this->database = (!empty($name))?$name:
$this->database;
            $url = $this->build_url();
            $result = $this->execute_query('PUT',
$url);
            return $result;
        }
        /*
        *Deletes the database currently stored in
$database
        */
        public function delete_db($name=null){
            $this->database = (!empty($name))?$name:
$this->database;
            $url = $this->build_url();
            $result = $this->execute_query('DELETE',
$url);
            return $result;
        }

        //returns database info object
        public function db_info(){
            $url = $this->build_url();
            $result = $this->execute_query('GET',
$url);
            return $result;
        }
        //returns the integer value that should be
unique by adding the total docs + deleted docs
        public function next_id(){
            $result = $this->db_info();
```

```
        $id = $result['doc_count']+
$result['doc_del_count'];
        return $id;
    }
    /*  Creates a document in whatever database is
currently set with the document ID that is set
    *
    *   $data is an array of key value pairs
    *
    */
    public function create_doc($data=null,
$doc_id=null){
        if(!empty($doc_id)&&!empty($data)){
            $url = $this->build_url($doc_id);
            $data = json_encode($data);
            $result = $this->execute_query('PUT',
$url, $data);
            if(isset($result['ok'])){
                return $result;
            }else{
                return false;
            }
        }else if(empty($doc_id)&&!empty($data)){
            $url = $this->build_url();
            $data = json_encode($data);
            $result = $this->execute_query('POST',
$url, $data);
            if(isset($result['ok'])){
                return $result;
            }else{
                return false;
            }
        }else  return array('error'=>'Please
provide a document ID and data');
    }


    /*
    * Gets a document given a document id and
optionally whatever parameters in the form of an
array (i.e. key="ben.guidarelli@newdigs.com")
```

```
    */
    public function get_doc($doc_id=null, $params
= null){
        $url = $this->build_url($doc_id, $params);
        $doc = $this->execute_query('GET', $url);
        if(isset($doc['_id'])){
            return $doc;
        }else{
            return false;
        }
    }


    public function
get_doc_by_view($design_doc=null, $view=null,
$key=null, $params=null){
        $params['include_docs'] = 'true';
        if($result = $this->get_view($design_doc,
$view, $key, $params)){
            return $result[0]['doc'];
        }else return false;

    }


    public function get_all($design=false){
        $url = $this->build_url('_all_docs',
array('include_docs'=>'true'));
        $doc = $this->execute_query('GET', $url);
        if(isset($doc['rows'])){
            if(!$design){
                $rows = array();
                foreach($doc['rows'] as $row){

                    if(strpos($row['id'],
'_design')===0){}
                    else $rows[] = $row['doc'];
                }
                return $rows;
            }
            return $doc['rows'];
```

```
        }else{
            return false;
        }
    }

    public function get_attributes($doc_id=null,
$fields=null, $params=null){
        $doc = $this->get_doc($doc_id, $params);
        $returning = array();
        foreach($fields as $field){
            $returning[$field] = $doc[$field];
        }
        return $returning;
    }
    public function get_attribute($doc_id=null,
$field=null, $params=null){
        $doc = $this->get_doc($doc_id, $params);
        return $doc[$field];
    }
    /*
    *Checks to see if a document exists and is
accessible with a head request. I don't remember
why I added it but I'll leave it anyway
    */
    public function exists($doc_id = null){
        if($doc_id == null) return;
        $result = $this->get_head($doc_id);
        if(strpos($result['status'], '200')){
            return true;
        }else {
            return false;
        }
    }


    /*
    * So this accepts an array which is the object
you want to add or merge with an existing document
essentially set_attributeS
    */
    public function add_obj($doc_id, $data){
```

```
        $doc = $this->get_doc($doc_id);

        $keys = array_keys($data);
        if(isset($doc[$keys[0]])){
            $doc = array_merge_recursive($doc,
$data);
        }else{
            $doc[$keys[0]]=$data[$keys[0]];
        }

        $url = $this->build_url($doc_id);
        $data = json_encode($doc);
        $revised = $this->execute_query('PUT',
$url, $data);
        if(isset($revised['ok'])){
            return $revised;
        }else{
            return false;
        }
    }
    public function set_attribute($doc_id=null,
$field=null, $value=null){
        $doc = $this->get_doc($doc_id, $params);
        $doc[$field] = $value;
        $result = $this->save($doc);
        return $doc[$field];
    }

    /*
    *This takes an array to be turned into an
object and PUTS it to the document id you give it.
This will completely overwrite whatever is there.
    */
    public function revise_doc($doc_id, $data){
        $url = $this->build_url($doc_id);
        $data = json_encode($data);
        $revised = $this->execute_query('PUT',
$url, $data);
        if(isset($revised['ok'])){
            return $revised;
        }else{
```

```
            return false;
        }
    }
    /*
    *This takes a document that still has its _id
and _rev and saves it
    */
    public function save($data=null){
        if($data==null) return false;
        if(isset($data['_id'])){  $url = $this-
>build_url($data['_id']);  $method = "PUT"; }
        else{ $url = $this->build_url(); $method =
"POST"; }
        $data = json_encode($data);
        $revised = $this->execute_query($method,
$url, $data);
        if(isset($revised['ok'])){
            return $revised;
        }else{
            return false;
        }
    }


    ///params is the query string array with
params['q'] being either a string or array of
key:values
    //if you want to make sure the key:value pair
is in the lucene doc prepend the key with a +
    //or a - if you want to make sure it is not in
the doc.
    public function text_search($design_doc,
$index, $params){
        $string = "_fti/_design/".$design_doc."/".
$index;
        $url = $this->build_url($string);
        $query_string = $this-
>build_fulltext_query($params);
        $url .= "?".rtrim($query_string);
        $result = $this->execute_query('GET',
$url);
```

```
            if(isset($result['rows'])){
                return $result;
            }else{
                return false;
            }
        }

        public function download_attachment($doc_id,
$attachment){
            $string = $doc_id."/".$attachment;
            $url = $this->build_url($string);
            $output = $this->execute_query('GET',
$url, null, false, true);
            return $output;
        }


        public function upload_attachment($doc_id,
$attachment_name, $data, $content_type){
            $rev = $this->get_head($doc_id, 'ETag');
            $rev = ltrim(rtrim($rev, "\""),"\"");
            $string = $doc_id."/".$attachment_name;
            $url = $this->build_url($string,
array('rev'=>$rev));
            $result = $this->execute_query('PUT',
$url, $data, false, true, $content_type);
            return $result;
        }

        /*
        *Deletes a document or a specific revision of
a document
        */
        public function delete_doc($doc_id=null,
$revision=null){
            if($revision=='ALL'){
                $url = $this->build_url($doc_id);
            }else{
                $rev = (!empty($revision))?
$revision:trim($this->get_head($doc_id, 'ETag'),
'"');;
```

```
            $url = $this->build_url($doc_id,
array('rev'=>$rev));
        }
        $result = $this->execute_query('DELETE',
$url, null);
        if(isset($result['ok'])){
            return $result;
        }else{
            return false;
        }
    }
        //takes a document id and the number of
revisions you want to keep and deletes all but
those
    public function delete_revs($doc_id=null,
$num=null){
        $data = array('revs_info'=>'true');
        $result = $this->get_doc($doc_id, $data);
        if(count($result['_revs_info'])>$num){
            for($x=$num;
$x<count($result['_revs_info']); $x++){
                $this->delete_doc($result['_id'],
$result['_revs_info'][$x]['rev']);
            }
        }
        $resulting = $this->get_doc($doc_id);
        return $resulting;
    }


    /*
    *Returns a specific revision of a document
    */
    public function get_rev($doc_id, $num){
        $data = array('revs_info'=>'true');

        $result = $this->get_doc($doc_id, $data);



        foreach($result['_revs_info'] as $value){
```

```
            $rev_num = (int)substr($value['rev'],
0, 1);
            if($rev_num===$num){
                if($value['status']=='available')
{
                    $dat =
array('rev'=>$value['rev']);
                    $frezult = $this-
>get_doc($doc_id, $dat);
                    return $frezult;
                }
            }
        }
    }



    //Takes the name of the desgin doc, the view
name, and an array of parameters to find
    public function get_view($design_doc=null,
$view=null, $params=null, $include_docs=false){
        $url = $this->build_url($this-
>build_view($design_doc, $view), $params);
        $result = $this->execute_query('GET',
$url);

if(isset($result['rows'])&&count($result['rows'])>
0){
            return $result['rows'];
        }else{
            return false;
        }
    }

    public function get_list($design_doc=null,
$list=null, $view=null, $params=null){
        $url = $this->build_url($this-
>build_view($design_doc, $view), $params);
        $result = $this->execute_query('GET',
$url);
```

```
if(isset($result['rows'])&&count($result['rows'])>
0){
                return $result['rows'];
            }else{
                return false;
            }
        }
        /*
        *Include an array of parameters like
array('since'=>25);
        */
        public function get_changes($params){
            $url = $this->build_url('_changes',
$params);
            $result = $this->execute_query('GET',
$url);
            return $result;
        }

        /*
        *Uses PHPs Build in curl functionality to make
requests to the couchdb when provided an HTTP
verb(method), the RESTful URL, and a json object
as the $data, the $header is for a head request
        */
        private function execute_query($method, $url,
$data=null, $header=false, $file=false,
$content_type=false){


            $ch = curl_init();


            curl_setopt($ch, CURLOPT_URL, $url);
            curl_setopt($ch, CURLOPT_CUSTOMREQUEST,
$method);
            curl_setopt($ch, CURLOPT_RETURNTRANSFER,
true);
            curl_setopt($ch, CURLOPT_BINARYTRANSFER,
$file);
```

```
        curl_setopt($ch, CURLOPT_NOBODY, $header);
        curl_setopt($ch, CURLOPT_HEADER, $header);
        if(!$file)   curl_setopt($ch,
CURLOPT_HTTPHEADER, array('Content-type:
application/json'));
        else curl_setopt($ch, CURLOPT_HTTPHEADER,
array('Content-type: '.$content_type));
        //if($this-
>options['password'])curl_setopt($ch,
CURLOPT_USERPWD, $this->options['username'].':'.
$this->options['password']);
        if(!empty($data))curl_setopt($ch,
CURLOPT_POSTFIELDS, $data);



        $result = curl_exec($ch);


        if($file){
            return $result;
        }

        if(!$header){
            $result = json_decode($result, true);
        }

        curl_close($ch);
        return $result;

    }


    /*
    *Makes a head request. Useful for just getting
a piece of the headers. Second favorite function
name.
    */
    private function get_head($doc_id,
$chunk=null){
        $url = $this->build_url($doc_id);
```

```
        $head = $this->execute_query('HEAD', $url,
null, true);
        $head = $this->http_parse_headers($head);
        if(isset($chunk)){
            return $head[$chunk];
        }else{
            return $head;
        }


    }

    /*
    *Builds the view section of the url
    */
    private function build_view($design_doc=null,
$view=null, $include_docs=false){
            $view = "_design/".$design_doc."/
_view/".$view;
            $view .= ($include_docs)?"?
include_docs=true":"";
            return $view;
    }


    /*
    *Builds the RESTful url of what youre trying
to access
    *Can accept an array for 'key' and url encodes
it correctly. I think. its worked so far for me.
    */
    private function build_url($doc_id=null,
$param=null){

        $url = 'http://'.$this->options['host'].
':'. $this->options['port'] . '/' . $this-
>database;
        $url .= ($doc_id)? '/'.$doc_id:'';

        if($param && count($param)>0){
            $url .= '?';
```

```
            foreach($param as $key=>$value){
                if($key=='key'||$key=='keys'||
$key=='startkey'||$key=='endkey'){
                    if(is_array($value)){
                        $url .= $key .'=[';
                        foreach($value as $v){
                            $url .=
'"'.urlencode($v).'",';
                        }
                        $url = rtrim($url,
",").']&';
                    }else{
                        if(false/
*is_numeric($value)*/){
                            $url .= $key.'='.
$value.'&';
                        }else{
                            $url.=
$key.'="'.urlencode($value).'"&';
                        }
                    }
                }else{
                    $url.= $key.'='.$value.'&';
                }
            }
            $url = rtrim($url, '&');
        }

        return $url;
    }


    private function build_fulltext_query($params)
{
        $query_string = "q=";
        if(isset($params['q'])){
            if(is_array($params['q'])){
                foreach($params['q'] as
$key=>$value){
                    if($value=="")continue;
```

```
                    $query_string .=
urlencode($key).":".urlencode($value)." ";
                }
            }else{
                $query_string .=
urlencode($params['q']);
            }
        }else return;


        $query_string .= "&";


        foreach($params as $key=>$value){
            if($key!="q")$query_string .=
$key."=".urlencode($value)."&";
        }


        $query_string = substr($query_string, 0,
-1);
        return $query_string;


    }


    /*
    *Separates the http headers into an
associative array (copied from php docs with a
couple changes)
    */
    private function
http_parse_headers($headers=false){
        if($headers === false){
            return false;
            }
        $headers = explode("\r\n",$headers);
        foreach($headers as $value){
            $header = explode(": ",$value);
            if($header[0]&&!isset($header[1])){
```

```
                    $headerdata['status'] =
$header[0];
                    }
                else if($header[0] &&
isset($header[1]) && $header[1]){
                    $headerdata[$header[0]] =
$header[1];
                    }
                }
            return $headerdata;
        }
}




?>
```