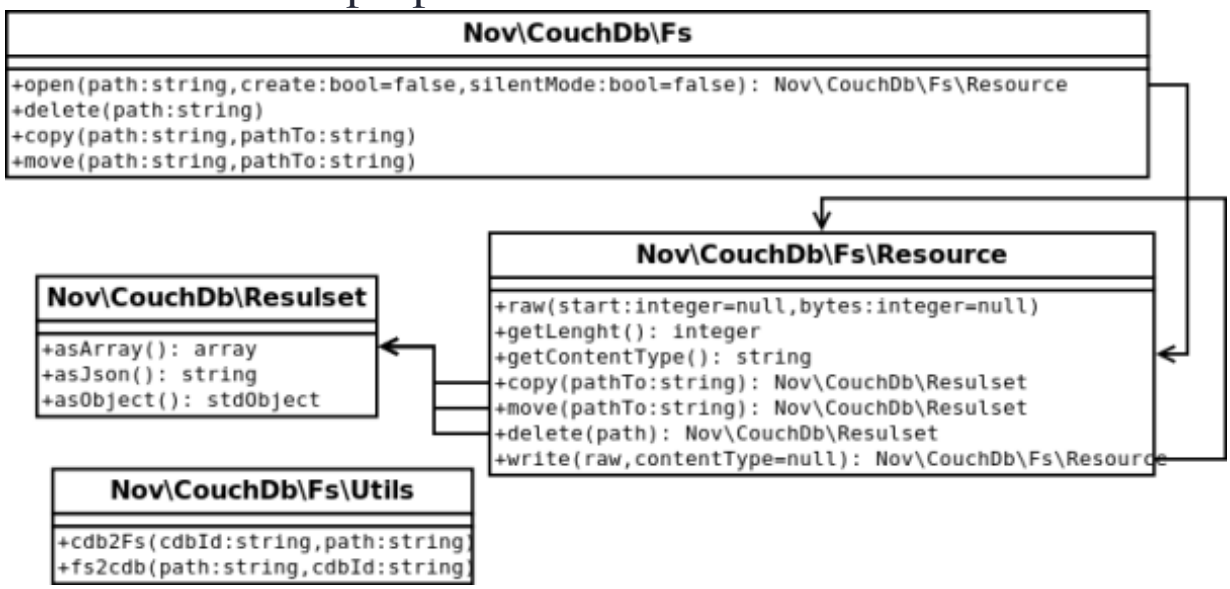
# Using CouchDb as a filesystem with PHP

**https://dzone.com/articles/using-couchdb-filesystem-php**

One of the problems I need to solve in my clustered PHP applications is where to store files. When I say files I'm not speaking about source code. I'm speaking about additional data files, such as download-able pdfs, logs, etc. Those files must be on every node of the cluster. One possible approach to the solution is to use a distributed filesystem, rsync or maybe use a file-server mounted on every node. Another solution may be the usage of CouchDb. CouchDb has two great features to meet or requirements with this problem. It allows us to store files as attachments and it also allows to perform a great and very easy multi-master replica system. The usage of CouchDB is pretty straightforward. It implements a RESTfull interface to perform every operations. So the only thing we need is a REST client. Zend Framework has a great one. We dont't really need a library. We can easily perform REST requests with the PHP's Curl's extension. I've created two libraries for working with CouchDb one is a low-level HTTP client (with curl) and another is higher level one (it uses the HTTP Client) for CouchDB operations. You can read two post about those libraries here and here.
Now I want to extend the features of my library. I want to

use CouchDB as file storage in PHP. Instead of using file functions (fopen, fwrite, fread, …) I want to use another ones and store/read files in CouchDB. For doing this I've refactored those two libraries into another one called Nov. I also have embraced namespaces so I will use them in the library. This means it's only available with PHP 5.3.
Here you are a summary of the library. That's not a complete UML graph. It's only a diagram with the main features only with educational purpose.




The best to show the library is with an example:
First I'm going to start with the basic usage of Nov\CouchDb library:

```
// Starting up the loader
require_once("Nov/Loader.php");
Nov\Loader::init();

use Nov\CouchDb;
$cdb = new CouchDb('localhost', 5984);
$cdb->db('users');
$nombre = $cdb->db('ris_users')->select('gonzalo')-
```

```
>asObject()->name;
$apellido = $cdb->db('ris_users')->select('gonzalo')-
>asObject()->surname;
echo "Hello {$nombre} {$apellido}.
";
```

To allow me the use of different CouchDb Databases and to put the Database configuration in one file. I use the following configuration class:

```
class NovConf
{
    const CDB1 = 'CDB1';
    const PG1  = 'PG1';

    public static $_dbs = array(
    self::PG1  => array(
            'driver'   => 'pgsql',
            'dsn'      =>
'pgsql:dbname=pg1;host=localhost',
            'username' => null
            'password' => null,
        ),
        self::CDB1  => array(
            'driver'   => 'couchdb',
            'host'     => 'localhost',
            'port'     => 5984,
            'protocol' => 'http',
            'username' => null,
            'password' => null,
        ),
    );
}
```

As you can see I use the same configuration file for my PDO drivers and CouchDb.

Now I can use:

```
require_once("Nov/Loader.php");
Nov\Loader::init();

use Nov\CouchDb;
```

```
$cdb = CouchDb::factory(NovConf::CDB1)->db('users');

try {
    $cdb->insert('xxx', array('name' => 'xxx'));
} catch (CouchDb\Exception\DupValOnIndex $e) {
    echo "Already created\n";
}

$data = $cdb->select('xxx')->asObject();
$cdb->update('xxx', array('name' => 'xxx1'));
$cdb->delete('xxx')->asObject();
require_once("Nov/Loader.php");
Nov\Loader::init();

use Nov\CouchDb;
$cdb = CouchDb::factory(NovConf::CDB1)->db('users');

try {
    $cdb->insert('xxx', array('name' => 'xxx'));
} catch (CouchDb\Exception\DupValOnIndex $e) {
    echo "Already created\n";
}

$data = $cdb->select('xxx')->asObject();
$cdb->update('xxx', array('name' => 'xxx1'));
$cdb->delete('xxx')->asObject();
```

And now finally the file storage part:

For storing the files I've taken one design decision. Every files will be stored into separate CouchDb document. That's means one file, one document. There's another possible approach. One CouchDb document can be one folder and store every files as attachments of this folder in the same document. But I prefer the idea of not to track folders. Only files. So each CouchDb document will have only one attachment.

That's an example of one document in CouchDb

```
{
   "_id": "/home/gonzalo/aasa.txt",
   "_rev": "2-48b501a81c38fd84a3e0351917e64135",
   "path": "/home/gonzalo",
   "_attachments": {
       "aasa.txt": {
           "stub": true,
           "content_type": "application/octet-stream",
           "length": 12,
           "revpos": 2
       }
   }
}
```

There's another usage script. Here we can see all the features together. We create files, update and delete them. Internally Nov\CouchDb\Fs uses a predefined CouchDb database called fs.

```
use Nov\CouchDb\Fs;
use Nov\CouchDb\Fs\Exception;
require_once ("Nov/Loader.php");
Nov\Loader::init();

echo "<pre>";
// create an instance from a factory method
$fs = Fs::factory(NovConf::CDB1);
// Now we're going to delete a file. If it doesn't
exists will throw a FileNotFound exception
try {
    $fs->delete("/home/gonzalo/aaa.txt");
} catch (Exception\FileNotFound  $e) {
    echo $e->getMessage() . "\n";
}
// Now we are going to create a file.
// the second parameter 'true' means if the file
doesn't exist will be created. Similar than 'r+'
try {
    $fs->open("/home/gonzalo/aaa.txt", true)
    ->write("asasasasasas", "application/octet-
stream");
```

```
} catch (Exception\FileNotFound $e) {
    echo $e->getMessage() . "\n";
} catch (Exception\WriteError $e) {
    echo $e->getMessage() . "\n";
} catch (Exception $e) {
    echo $e->getMessage() . "\n";
}
// We open the file
$res = $fs->open("/home/gonzalo/aaa.txt");

// we can get the length and the content type
echo $res->getLenght() . "\n";
echo $res->getContentType(). "\n";
// We move it to another location
$to = "/another/location";
$res->move($to);

$res = $fs->open($to);
// we flush the file to the browser
echo $res->raw();

// finally we delete it
$res->delete();
echo "</pre>";
```

I've also created an extra class to allow to dump files from filesystem to CouchDb and vice-versa.

```
require_once ("Nov/Loader.php");
Nov\Loader::init();
echo "<pre>";
// from filesystem to couchdb
\Nov\CouchDb\Fs\Utils::fs2cdb("/path/from/",
NovConf::CDB1);
// from couchdb to filesystem
\Nov\CouchDb\Fs\Utils::cdb2fs(NovConf::CDB1, "/path/
to/");
echo "</pre>";
```

And that's all. You can download the source code with the examples here. The examples are under document_root/tests/couchdb/ folder. Remember you will need PHP5.3.

Source: http://gonzalo123.wordpress.com/2010/08/30/using-couchdb-as-filesystem-with-php/