PHP Simple REST Framework :

This document describes structure and functionality of the Framework Classes.

This framework can be used to create "JSON Api web service»

Structure of the REST requests :

http://server.com/api/users/getlist?fget=1 -

1 http://server.com - domain of the service

2 /api/ - subDirectory where framework files are located

3 /users/ - Name of the Class .

4 /getlist - get method name in the class

5 fget - some get variables

Structure of the Framework :

-- Root

-- api

--commands

--Helpers

--media

--models

--tmp

--Bootstrap.php

--config.php

--index.php

--Observer.php

--REquest.php

--Response.php

--serviceDesc.txt

--.htaccess

1 Commands - This Directory contains all Classes required for your service .

Each classes should be named exactly as the name of the method for example Users.php

Will be accessible with the followed request

http:// server.com/api/Users/name_of_the_method inside Users class that returns Response object.

Each Class should be extended From Abstract method Class.

2 Helpers - This folder contains static Helpers classes that you can use in your commands or models

3 media - contains all media uploaded file

4 tmp - contains temporary files

5 models - Contains all models Classes extended from base Abstract Model .

. Methods Overview:

Commands Abstract Methods :

requireToken() ; - This method need to be called inside methods that requires secure token $_GET[“token”] ; see Quick start section :

allowCache(); -- this method should be called inside methods that allowing cache output result in memcache ;

disableCache(); -- disable cache for current method ;

setError($msg) ; -- Creates Error Json response with error string ;

Execute(\Request $req = null) ; -- Default method will be executed on each Commands Class ;

hasResponse() ; -- returns true or false if current Command has Reponse.

Model Abstract Methods:

public function setTable($table) Set Name of the Table For model ;

public function delete($id) -- Remove record By id in current table ;

public function setSql($sql) -- Set Raw SQl;

public function exec() -- Execute Raw SQL ;

public function selectAll()-- Select all from raw SQL request ;

public function load() -- Load Single Object Record from sql raw request ;

public function loadList() -- Load list of object from raw Request ;

public function selectByAll($field, $v) -- Select record set Where field = v ;

public function generalInsert($data = []) -- insert Key – value into the table .

public function updateBypost($post = []) -- Update record "If dataset contains id key "

public function createToken($data_users = []) -- Create Secure Token with Custom Data returns hash string ;

public function getFromToken($name , $var) -- Get data from token by hash and var name ;

Request Method

public function contains($name) -- return true or false if request cdontains variable

public function get($name) -- get request get or post variable ;

Response Object methods

public function __construct($status = true, $error = false , $error_m = "") – default constructor ;

public function add($key , $val=null) -- add key value into data response array ;

Observer Methods:

public function UploadFile($path = '', $prefix = '') -- upload single File

and returns the name of the file

$path – default upload path FROM BASE_DIR ;

$prefix prefix name of the file ;

public function UploadFiles($path = '', $prefix = '') -- Upload Multiply Files and return array of file names ;

public function createThumbnail($filename, $prefix = '_small') – Create image Thumbnail

public function sendFile($filename = 'file.csv', $data = '') -- Send file to the end user with headers ;

public function auto_link($str) -- Auto link string ;

public function saveHTML($str) – strip some html tags ;

public function compress ($buffer) -- compress string;

Quick Start :

1 Grab your copy From GitHub . it requires php 5.4 or later ;

2 Place it on your web server for example in /api/directory ;

3 Edit .htaccess Change RewriteBase

4 Open And Edit config.php

Edit Database Settings :

HOST mysql server

USER mysql username

PASS mysql password

DATABASE name of database

DEVELOP - true - false ; display or not errors

START_PATH - Integer value for Parsing url if your service is in subdirectory the start path will be start from 3 else if root directory you should change it to 2

SITE_NAME - Name of the service will be display in each response

VERSION - string will be output in Json Response

FORMAT - “JSON” / “XML” Format of the output ;

SMTP_HOST SMTP_USER SMTP_PASS NOTIFY_EMAIL - SMTP settings For Mail Helper

5 Test it / open url in your browser http://your.com/api will be output default response with Timestamp

CREATE SIMPLE "HELLO WORLD"

When after that we finished configuration, let's create simple hello World Service.

Lets start with creating a simple table in our database.

Create Table "Users" with fields : id, username , email ;

In models directory create file Users.php in that file create Class Users extended from _Abstract

You should get something like :

```php
<?php
namespace models;

class Users extends _Abstract{




}
```

and create constructor :

```php
        public function __construct(){
             parent::__construct() ;
             $this->setTable("Users") ;  // set Table For that
model ;


        }
```

Ok we have created model For Users model, time Create our Actions

Lets create action that will have methods for register new user, login, and get users list;

In /Api/commands / create class

Users file name Users.php

```php
<?php
namespace commands;

class Users extends _Abstract{


     public function hasResponse(){
          return true ;
     }









}
```

// ok let's create Business logic ;

in our model Users add method That will insert new user

```php
public function insertUser($username , $email){
                $id = $this->generalInsert([
                     "username"=>$username,
                     "email"=>$email ,
                 ]);
                return $id  ;
           }
```

in commands/Users create method that will encapsulate login register :

```php
     public function register(\Request $req = null ){
          if  ($req->contains("username") && $req->contains("email")){
                $model =  new  Users() ;

                // Create user and add id  in response
                $this->responseObject->add("user_id",
                          $model->insertUser( $req->get("username") , $req->get("email")));
```

```
            } else{
                  $this->setError("username ,email required") ;
            }
            return $this->responseObject ;
      }
```

// Great now we can create new users in our table :

like this

http://server.com/api/users/register?username=some&email=some

now it's time to create login method

in Command/users create method login

```
      public function login(\Request $req = null){
            $model =  new  Users() ;
            $username  = $req->get("username") ;

            if ($username){
                  // try to find out user
                  $id =  $model->login($username);
                  if ($id) {
                        $this->responseObject->add("user_id", $id) ;
                        $this->responseObject->add("token", $model-
>createToken()) ;

                  }else{
                        $this->setError("User Not  Found") ;
                  }

            }else{
                  $this->setError("Please enter username") ;
            }

            return $this->responseObject ;
      }
```

ok for login we just need to call

http://server.com/users/login?username=some

note this is the test , so there is no password etc.

anyway , on success it will return :

user_id , token ;

token we will gonna need that in our last method called “get users List” :

Lets Create method in our controller that will return list of users for logged users ;

```
public function getlist(\Request $req = null){
      $model =  new Users();
      $this->requireToken() ; // Token Required For that method ;
       $all = $model->selectAll();
       $this->responseObject->add("all", $all) ;
     return $this->responseObject;
}
```

This method should be called :

http://server.com/api/users/getlist?token=…… . token string

See full example in repository: