Php things

Connection

Pdo or mysqli for POO, MYSQLI for procedural

PDO:

Class dbConnection = {

$username=’juan’;

$password = ‘1234’;

Public function connect($username, $password){

$username = $this->username;

$password = $this->password;

$conn = new PDO(‘mysql:host=localhost,dbname=dbname’,$username,$password);

If(!conn){

Echo ‘error’;

Exit();

}return $conn;

}

}

Class executeConnection extends dbConnection{

public function search(param1){

try{

$sql = ‘SELECT \* FROM table WHERE name = ?’;

$stmt = $this->connect()->prepare($sql);

$stmt->execute($param1);

If($stmt->rowCount()===0){

Echo ‘error’;

}else{

While($result = $stmt->fetch()){

Echo ‘results’;

}

}

}catch(PDOException $e){

Echo e->getMessage();

}

}

}

//another cruds operations will need binds

1) stmt\_init()

2) stmt\_prepare()

3) stmt\_bindParams()

4)stmt\_execute()

Public function addThing($param1,$param2, $param2){

$stmt = $this->connect()->prepare(function\_that\_add());

$stmt->bindParam(‘:param1’,$param1,PDO::PARAM\_STR);

$stmt->bindParam(‘:param2’,$param2,PDO::PARAM\_INT);

$stmt->bindParam(‘:param3’,$param3,PDO::PARAM\_STR);

$stmt->execute();

}

/\*execute the function throught a opc ajax call

If($opc === ‘something’){

Call function();

}

//Js ajax custom function

function setGetRefresh(link,responseItem,objeto){

    var xhttp = new XMLHttpRequest();

    xhttp.onreadystatechange = function() {

      if (this.readyState == 4 && this.status == 200) {

        document.querySelector(responseItem).innerHTML = this.responseText

      }

    };

    xhttp.open("POST", link, true);

    xhttp.setRequestHeader("Content-type", "application/json");

    xhttp.send(JSON.stringify(objeto));

}

// send to search

searchDrink.addEventListener('click',(e)=>{

let drinkName = document.querySelector('input[name="drink"]');

let drinkTag = document.querySelector('select[name="tag"]');

e.preventDefault();

let obj={

    drink:drinkName.value,

    tag:drinkTag.value,

    opc:'get\_drink'

}

setGetRefresh('./execute.php','.column-two',obj);

});

**PHP API**

Create a json file with the data

[{"id":"0","name":"Pedrin","description":"tttt"},{"id":"1","name":"Juan Ramirez","description":"rrrr"}]

Create a class that contains al the fucntions

Class task{

 private $id;

    private $name;

    private $description;

public function \_\_construct($id,$name,$description){

    $this->id = $id;

    $this->name = $name;

    $this->description = $description;

}

Create function

Public function saveUserTask(){

$content = file\_get\_contents(‘task.json’);

$task = json\_decode($content,true);

$task[]=array(

‘id’=>$this->id,

‘name’=>$this->name,

‘description’=>$this->description

);

$file = fopen(‘task.json’,’’w);

Fwrite($file,json\_encode($task));

Fclose($file);

}

Public function getUsersTask(){

$content = file\_get\_contents(‘task.json’);

Echo $content;

}

Public function getOneUserTask($index){

$content = file\_get\_contents(‘task.json’);

$user = json\_decode($content,true);

Echo json\_encode($user[$index]);

}

Update and delete

public function updateTask($index){

        $content = file\_get\_contents('task.json');

        $users = json\_decode($content,true);

        //$user = $users[$index];

        $user = array(

            'id'=>$this->id,

            'name'=>$this->name,

            'description'=>$this->description

        );

        $users[$index] = $user;

        $file = fopen('./task.json','w');

        fwrite($file,json\_encode($users));

        fclose($file);

        echo json\_encode($user);

    }

    public static function deleteTask($index){

        $content = file\_get\_contents('task.json');

        $users = json\_decode($content,true);

        array\_splice($users,$index,1);

        $file = fopen('./task.json','w');

        fwrite($file,json\_encode($users));

        fclose($file);

    }

}

Task file

Include the file that contains the class

Include the header application type and request method

    header('Content-type:application/json');

    //echo 'informacion' .file\_get\_contents('php://input');

    $serverMethod = $\_SERVER['REQUEST\_METHOD'];

Switch between methods to execute functions

switch($serverMethod){

        case 'POST':

            $\_POST = json\_decode(file\_get\_contents('php://input'),true);

            $task = new Task($\_POST['id'],$\_POST['name'],$\_POST['description']);

            $task->saveUserTask();

            echo 'obtenido los datos';

            $resultado['mensaje'] = 'Guardado los datos del usuario '. json\_encode($\_POST);

            echo json\_encode($resultado);

        break;

        case 'GET':

            if(isset($\_GET['id'])){

                Task::getOneUserTask($\_GET['id']);

            }else{

                Task::getUserTask();

            }

        break;

        case 'PUT':

            $\_PUT = json\_decode(file\_get\_contents('php://input'),true);

            $task = new Task($\_PUT['id'],$\_PUT['name'],$\_PUT['description']);

            $task->updateTask($\_GET['id']);

            $resultado['mensaje'] = 'editando los datos del usuario '.$\_GET['id'].'<br>'.

                                    'Editado el usuario '. json\_encode($\_PUT);

            echo json\_encode($resultado);

        break;

        case 'DELETE':

            $\_DELETE = json\_encode(file\_get\_contents('php://input'),true);

            Task::deleteTask($\_GET['id']);

            $resultado['mensaje'] = 'Eliminado el usuario '.$\_GET['id'];

            echo json\_encode($resultado);

        break;

        default: break;

    }

When not params needed use magic methos :: to call the function without construct or initializing

Include axios

<script src="https://cdnjs.cloudflare.com/ajax/libs/axios/0.20.0/axios.min.js" integrity="sha512-quHCp3WbBNkwLfYUMd+KwBAgpVukJu5MncuQaWXgCrfgcxCJAq/fo+oqrRKOj+UKEmyMCG3tb8RB63W+EmrOBg==" crossorigin="anonymous"></script>

<script src="./js/app.js"></script>

Axios it’s almost like ajax, use it like this

    axios({

            method:'POST',

            url:url,

            responseType:'json',

            data:userTask

        }).then(res=>{

            console.log(res);

            getUser();

            document.querySelector('input[name="name"]').value='';

            document.querySelector('input[name="desc"]').value='';

        }).catch(err=>{

            console.log(err);

        })

The param data its to past items to the method

PHPUNIT

Windows

Globally installing the PHAR involves the same procedure as manually [installing Composer on Windows](https://getcomposer.org/doc/00-intro.md" \l "installation-windows" \t "_top):

1. Create a directory for PHP binaries; e.g., C:\bin
2. Append **;C:\bin** to your PATH environment variable ([related help](http://stackoverflow.com/questions/6318156/adding-python-path-on-windows-7" \t "_top))
3. Download [https://phar.phpunit.de/phpunit-6.5.phar](https://phar.phpunit.de/phpunit-6.5.phar" \t "_top) and save the file as C:\bin\phpunit.phar
4. Open a command line (e.g., press **Windows**+**R** » type **cmd** » **ENTER**)
5. Create a wrapping batch script (results in C:\bin\phpunit.cmd):
6. C:\Users\username> **cd C:\bin**
7. C:\bin> **echo @php "%~dp0phpunit.phar" %\* > phpunit.cmd**
8. C:\bin> **exit**
9. Open a new command line and confirm that you can execute PHPUnit from any path:
10. C:\Users\username> **phpunit --version**

PHPUnit x.y.z by Sebastian Bergmann and contributors.