Code Your Own PHP MVC Framework in 1 Hour

Chris_Yu, 24 Feb 2016

★★★★★ 4.83 (14 votes) Rate this:

To code our own MVC framework from scratch

Introduction

MVC architectural pattern is almost in everywhere today, whether you are working on Java, C#, PHP, iOS projects. This might not be 100% exact, but PHP community has the most amount of MVC frameworks. Today you might be using Zend, tomorrow on another project you might have to change to Yii or Laravel or CakePHP. If you are new to MVC frameworks and you just download one from the official website, you might feel overwhelmed when you look at the framework's source code, yes, it is complex, as these popular frameworks are not written in a month - they are published, refined, tested again and again, and the functionalities are added constantly. So from my experience, knowing the core design methodologies of MVC framework is critical, otherwise you might feel that you will have to learn another framework and another framework again and again when you get new projects using new frameworks.

The best way to understand MVC is to write you own MVC framework from scratch! In this series of articles I am going to show you how to code one, so that you might get to understand why certains things happen that way in a framework.

MVC architectural pattern

M: Model

V: View

C: Controller

The core concept of MVC is to separate business logic from displaying(the View part). First let me explain the whole workflow of an HTTP request & HTTP response. For example, we have a commerce website and we want to add a certain product. A very simple URL would be look like this:

http://bestshop.com/index.php?p=admin&c=goods&a=add

http://bestshop.com is the domain name or base URL;

p=admin means the Platform is admin panel, or the Backend site of the system. We also have a Frontend site, which is public to the world(in this case, it would be p=public)

c=goods&a=add means this URL requests the ‘goods’ Controller’s add action method.

Front Controller Design Pattern

What is index.php in the above example? This file is called ‘Front Controller’ in PHP’s MVC frameworks. The name is usually index.php, but you can name it something else(few people do that thought…) One of the function of this index.php file is, it works as the single entry point for any HTTP requests, that is, no matter what resource you request in the URL, it will all go into this index.php file first. But why? How does the magic happen? Front Controller design pattern is implemented in PHP using Apache HTTP server’s distribution configuration file called .htaccess. In this file, we can tell the Apache HTTP server to redirect all requests to index.php using its rewrite module. You can code similar to this:

Hide Copy Code

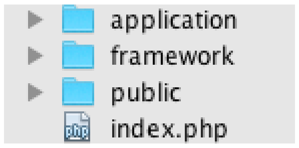
```
<IfModule mod_rewrite.c>


   Options +FollowSymLinks

   RewriteEngine on


   # Send request via index.php

   RewriteCond %{REQUEST_FILENAME} !-f

   RewriteCond %{REQUEST_FILENAME} !-d

   RewriteRule ^(.*)$ index.php/$1 [L]

</IfModule>
```

This configuration file is very powerful and when you change it, you don’t need to restart Apache. If you change Apache’s other configuration files, you need to restart it, the reason is that for other Apache configuration files, Apache only will read them when it’s started, that’s why any change requires a restart(Apache is mainly written in C by the way). But for .htaccess distribution configuration file, no restart is needed for any change.

Simply put, index.php will also do the proper initialization for the framework and it will route the request to the proper Controller(goodsController in above example) and an action method(member function) within that Controller class.

Our MVC Framework’s Structure

So let’s create our frameworks structure.



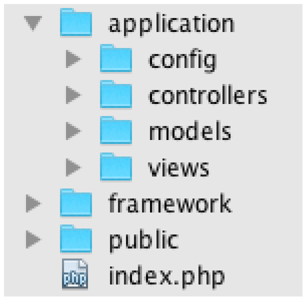
application directory is the web app’s directory;

framework directory is for the framework itself;

public directory is to store all the public static resources like html, css and js files.

index.php is the ‘entry-point’ file, the front controller.

Now within application folder, we create these subfolders:



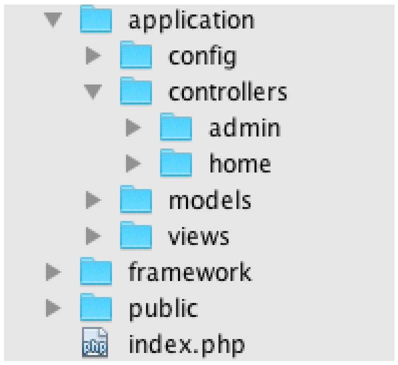
config - stores the app’s configuration files

controllers - this is for all app’s Controller classes

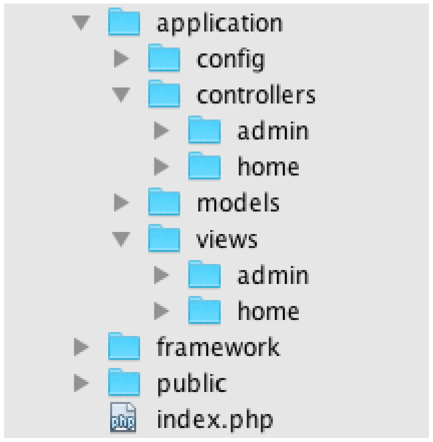
model - this is for all app’s Model classes

view - this is for all app’s View classes

Now within the application/controllers folder, we will create two folders for the frontend and backend platforms:

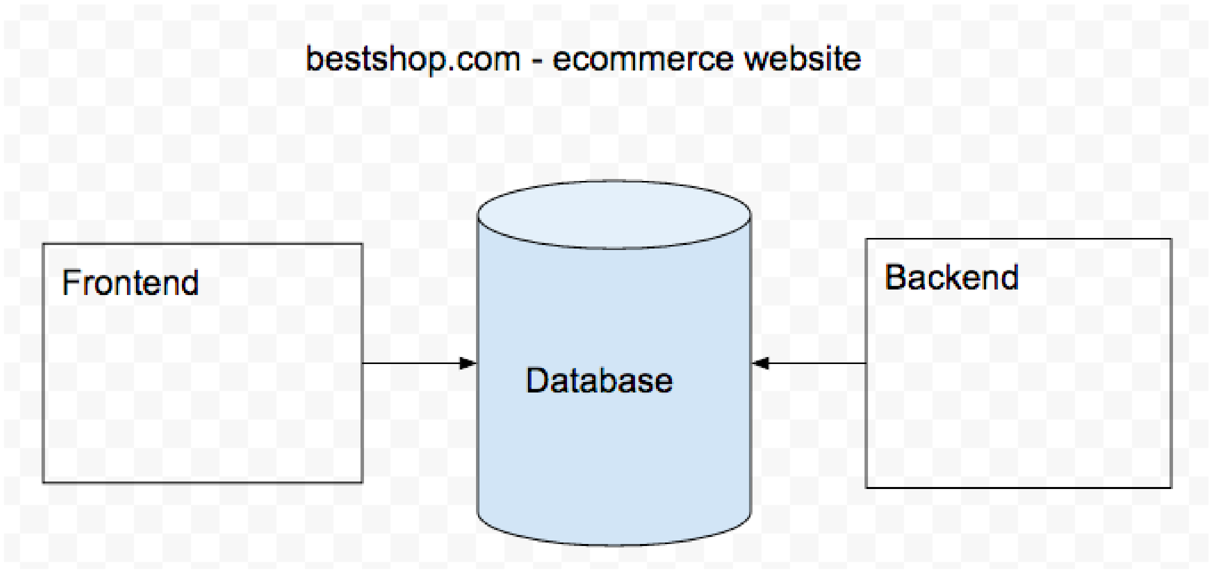


And in the view folder, the same for frontend and backend:



As you can see, within the application folder, we created frontend and backend subfolder in controllers and views folder, as our application has a frontend site and a backend site. But why we did not do the same in the models folder?

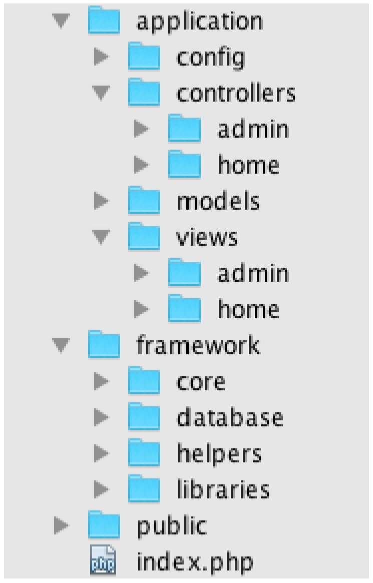
Well, the reason here is, normally for a web app:



Frontend and Backend can be two different ‘websites’, but the are CRUD the same database, that’s why an internal user updated a product’s price, the price is immediately show on the frontend public page - backend and frontend share the same database/tables.

So that’s why both backend and frontend can share a set of Model classes, and that’s why we didn’t create separate folders in the models folder.

Now let’s move on to the framework directory, some frameworks name this folder using the framework’s name, say ‘symfony’. In this folder, we quickly create these subfolders first:



```
application
    config
    controllers
        admin
        home
    models
    views
        admin
        home
framework
    core
    database
    helpers
    libraries
public
index.php
```

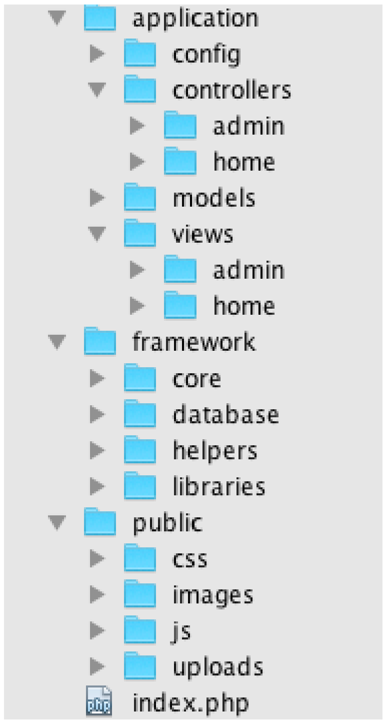
core - it will store the framework's core classes

database - database related classes, such as database driver classes

helpers - help/assistant functions

libraries - for class libraries

Now move on to the public folder, we create these subfolders:



```
application
    config
    controllers
        admin
        home
    models
    views
        admin
        home
framework
    core
    database
    helpers
    libraries
public
    css
    images
    js
    uploads
index.php
```

css - for css files

images - for images files

js - for javascript files

uploads - for uploaded files, such as uploaded images

Ok, so far this is our mini MVC framework’s structure.

Framework’s core class

Now under framework/core folder, we’ll create the framework’s first class - Framework.class.php in framework/core folder

Hide Copy Code

```
// framework/core/Framework.class.php

class Framework {


   public static function run() {

       echo "run()";

   }
```

we created a static function called run(). Now test it in index.php:

Hide Copy Code

```
<?php


require "framework/core/Framework.class.php";


Framework::run();
```

You can see the result in browser(how to config your virtual host is skipped here). Normally this static function is called run() or bootstrap(). Within this function, we can do 3 main things here, as shown in the code below:

Hide Shrink Copy Code

```
class Framework {


   public static function run() {

//         echo "run()";

       self::init();

       self::autoload();

       self::dispatch();

   }


   private static function init() {

   }
```

```

    private static function autoload() {

    }

    private static function dispatch() {

    }
}
```

Initialization

Here is the code for the init() method:

Hide Shrink Copy Code

```
// Initialization
private static function init() {
    // Define path constants
    define("DS", DIRECTORY_SEPARATOR);
    define("ROOT", getcwd() . DS);
    define("APP_PATH", ROOT . 'application' . DS);
    define("FRAMEWORK_PATH", ROOT . "framework" . DS);
    define("PUBLIC_PATH", ROOT . "public" . DS);

    define("CONFIG_PATH", APP_PATH . "config" . DS);
    define("CONTROLLER_PATH", APP_PATH . "controllers" . DS);
    define("MODEL_PATH", APP_PATH . "models" . DS);
    define("VIEW_PATH", APP_PATH . "views" . DS);

    define("CORE_PATH", FRAMEWORK_PATH . "core" . DS);
    define('DB_PATH', FRAMEWORK_PATH . "database" . DS);
    define("LIB_PATH", FRAMEWORK_PATH . "libraries" . DS);
    define("HELPER_PATH", FRAMEWORK_PATH . "helpers" . DS);
    define("UPLOAD_PATH", PUBLIC_PATH . "uploads" . DS);

    // Define platform, controller, action, for example:
```

```
    // index.php?p=admin&c=Goods&a=add

    define("PLATFORM", isset($_REQUEST['p']) ? $_REQUEST['p'] : 'home');

    define("CONTROLLER", isset($_REQUEST['c']) ? $_REQUEST['c'] : 'Index');

    define("ACTION", isset($_REQUEST['a']) ? $_REQUEST['a'] : 'index');


    define("CURR_CONTROLLER_PATH", CONTROLLER_PATH . PLATFORM . DS);

    define("CURR_VIEW_PATH", VIEW_PATH . PLATFORM . DS);


    // Load core classes

    require CORE_PATH . "Controller.class.php";

    require CORE_PATH . "Loader.class.php";

    require DB_PATH . "Mysql.class.php";

    require CORE_PATH . "Model.class.php";


    // Load configuration file

    $GLOBALS['config'] = include CONFIG_PATH . "config.php";


    // Start session

    session_start();

}
```

From the comments you can see the purpose of each steps.

Autoloading

We don’t want to manually code include or require for a class file what we need in every script in the project, that’s why PHP MVC frameworks have this autoloading feature. For example, in Symfony, if you put your own class file under ‘lib’ folder, then it will be auto loaded. Magic? No, there is no magic. Let’s implement our autoloading feature in our mini framework.

Here we need to use a PHP built-in function called spl_autoload_register

Hide Shrink ▲ Copy Code

```
// Autoloading

private static function autoload(){

    spl_autoload_register(array(__CLASS__,'load'));

}
```

```
// Define a custom load method

private static function load($classname){


    // Here simply autoload app’s controller and model classes

    if (substr($classname, -10) == "Controller"){

        // Controller

        require_once CURR_CONTROLLER_PATH . "$classname.class.php";

    } elseif (substr($classname, -5) == "Model"){

        // Model

        require_once  MODEL_PATH . "$classname.class.php";

    }

}
```

Every framework has a name conversion, ours is no exception. For a controller class, it should be xxxController.class.php, for a model class, it should be xxxModel.class.php. Why for a new framework you come across, you must follow its naming convention? Autoloading is one of the reasons.

Routing/Dispatching

Hide Copy Code

```
// Routing and dispatching

private static function dispatch(){

    // Instantiate the controller class and call its action method

    $controller_name = CONTROLLER . "Controller";

    $action_name = ACTION . "Action";

    $controller = new $controller_name;

    $controller->$action_name();

}
```

In this step, index.php will dispatch the request to the proper Controller::Action() method. It’s very simple here just for an example.

Base Controller class

There is always a base controller class(or several) in the framework's core classes. For example, In Symfony it's called sfActions; in iOS it's called UIViewController. Here we will just name it Controller, the file name is Controller.class.php

Hide Shrink Copy Code

```
<?php

// Base Controller

class Controller{

    // Base Controller has a property called $loader, it is an instance of Loader class(introduced later)

    protected $loader;


    public function __construct(){

        $this->loader = new Loader();

    }


    public function redirect($url,$message,$wait = 0){

        if ($wait == 0){

            header("Location:$url");

        } else {

            include CURR_VIEW_PATH . "message.html";

        }


        exit;

    }

}
```

Base Controller has a property called $loader, it is an instance of Loader class(introduced later). Please note the phrase ' it is an instance of Loader class' -- preciously speaking, $this->loader is a reference variable which reference/points to an instance of Load class. We don't talk more about this here, but this is actually a very important concept. I have met some PHP developers who believe after this statement:

Hide Copy Code

```
$this->loader = new Loader();
```

$this->loader is an object. No, it's a reference. This term starts from Java, before Java, it's called pointer in C++ or Objective C. Reference is an encapsulated pointer type. For example, in iOS(Objective-C) we create an object using:

Hide Copy Code

```
UIButton *btn = [UIButton alloc] init];
```

Loader class

In framework.class.php, we have already implemented application's controller and model class' autoloading. But how to load classes in the framework directory? Here we can create a new class called Loader, it will be used to load the framework's classes and functions. When we need to load a framework's class, just call this Loader class's method.

Hide Copy Code

```
class Loader{

    // Load library classes

    public function library($lib){

        include LIB_PATH . "$lib.class.php";

    }


    // loader helper functions. Naming conversion is xxx_helper.php;

    public function helper($helper){

        include HELPER_PATH . "{$helper}_helper.php";

    }

}
```

Implementing Model

We will implement Model in the simplest way, by creating two class files:

Mysql.class.php - this class is under framework/database, it is to encapsulate database connection and some basic SQL query methods.

Model.class.php - this is the Base Model class, it contains methods for all kinds of CRUD

Hide Shrink ▲ Copy Code

```
<?php

/**

*================================================================

*framework/database/Mysql.class.php
```

```
*Database operation class

*================================================================

*/

class Mysql{

    protected $conn = false;  //DB connection resources

    protected $sql;           //sql statement


    /**

     * Constructor, to connect to database, select database and set charset

     * @param $config string configuration array

     */

    public function __construct($config = array()){

        $host = isset($config['host'])? $config['host'] : 'localhost';

        $user = isset($config['user'])? $config['user'] : 'root';

        $password = isset($config['password'])? $config['password'] : '';

        $dbname = isset($config['dbname'])? $config['dbname'] : '';

        $port = isset($config['port'])? $config['port'] : '3306';

        $charset = isset($config['charset'])? $config['charset'] : '3306';


        $this->conn = mysql_connect("$host:$port",$user,$password) or die('Database connection
error');

        mysql_select_db($dbname) or die('Database selection error');

        $this->setChar($charset);

    }

    /**

     * Set charset

     * @access private

     * @param $charset string charset

     */

    private function setChar($charest){

        $sql = 'set names '.$charest;

        $this->query($sql);

    }
    /**

     * Execute SQL statement

     * @access public

     * @param $sql string SQL query statement

     * @return $result，if succeed, return resrouces; if fail return error message and exit

     */

    public function query($sql){

        $this->sql = $sql;

        // Write SQL statement into log

        $str = $sql . "  [". date("Y-m-d H:i:s") ."]" . PHP_EOL;

        file_put_contents("log.txt", $str,FILE_APPEND);

        $result = mysql_query($this->sql,$this->conn);


        if (! $result) {

            die($this->errno().':'.$this->error().'<br />Error SQL statement is '.$this->sql.'<br
/>');

        }

        return $result;

    }

    /**

     * Get the first column of the first record

     * @access public

     * @param $sql string SQL query statement

     * @return return the value of this column

     */

    public function getOne($sql){

        $result = $this->query($sql);

        $row = mysql_fetch_row($result);

        if ($row) {

            return $row[0];

        } else {

            return false;

        }
    }

    /**

     * Get one record

     * @access public

     * @param $sql SQL query statement

     * @return array associative array

     */

    public function getRow($sql){

        if ($result = $this->query($sql)) {

            $row = mysql_fetch_assoc($result);

            return $row;

        } else {

            return false;

        }

    }

    /**

     * Get all records

     * @access public

     * @param $sql SQL query statement

     * @return $list an 2D array containing all result records

     */

    public function getAll($sql){

        $result = $this->query($sql);

        $list = array();

        while ($row = mysql_fetch_assoc($result)){

            $list[] = $row;

        }

        return $list;

    }

    /**

     * Get the value of a column

     * @access public
     * @param $sql string SQL query statement

     * @return $list array an array of the value of this column

     */

    public function getCol($sql){

        $result = $this->query($sql);

        $list = array();

        while ($row = mysql_fetch_row($result)) {

            $list[] = $row[0];

        }

        return $list;

    }



    /**

     * Get last insert id

     */

    public function getInsertId(){

        return mysql_insert_id($this->conn);

    }

    /**

     * Get error number

     * @access private

     * @return error number

     */

    public function errno(){

        return mysql_errno($this->conn);

    }

    /**

     * Get error message

     * @access private

     * @return error message

     */

    public function error(){
        return mysql_error($this->conn);

    }

}
```

Here is the Model.class.php

Hide Shrink ▲ Copy Code

```
<?php

// framework/core/Model.class.php

// Base Model Class

class Model{

    protected $db; //database connection object

    protected $table; //table name

    protected $fields = array();  //fields list

    public function __construct($table){

        $dbconfig['host'] = $GLOBALS['config']['host'];

        $dbconfig['user'] = $GLOBALS['config']['user'];

        $dbconfig['password'] = $GLOBALS['config']['password'];

        $dbconfig['dbname'] = $GLOBALS['config']['dbname'];

        $dbconfig['port'] = $GLOBALS['config']['port'];

        $dbconfig['charset'] = $GLOBALS['config']['charset'];


        $this->db = new Mysql($dbconfig);

        $this->table = $GLOBALS['config']['prefix'] . $table;

        $this->getFields();

    }

    /**

     * Get the list of table fields

     *

     */

    private function getFields(){

        $sql = "DESC ". $this->table;

        $result = $this->db->getAll($sql);

        foreach ($result as $v) {
            $this->fields[] = $v['Field'];

            if ($v['Key'] == 'PRI') {

                // If there is PK, save it in $pk

                $pk = $v['Field'];

            }

        }

        // If there is PK, add it into fields list

        if (isset($pk)) {

            $this->fields['pk'] = $pk;

        }

    }

    /**

     * Insert records

     * @access public

     * @param $list array associative array

     * @return mixed If succeed return inserted record id, else return false

     */

    public function insert($list){

        $field_list = '';  //field list string

        $value_list = '';  //value list string

        foreach ($list as $k => $v) {

            if (in_array($k, $this->fields)) {

                $field_list .= "`".$k."`" . ',';

                $value_list .= "'".$v."'" . ',';

            }

        }

        // Trim the comma on the right

        $field_list = rtrim($field_list,',');

        $value_list = rtrim($value_list,',');

        // Construct sql statement

        $sql = "INSERT INTO `{$this->table}` ({$field_list}) VALUES ($value_list)";

        if ($this->db->query($sql)) {
            // Insert succeed, return the last record’s id

            return $this->db->getInsertId();

            //return true;

        } else {

            // Insert fail, return false

            return false;

        }


    }

    /**

     * Update records

     * @access public

     * @param $list array associative array needs to be updated

     * @return mixed If succeed return the count of affected rows, else return false

     */

    public function update($list){

        $uplist = ''; //update fields

        $where = 0;   //update condition, default is 0

        foreach ($list as $k => $v) {

            if (in_array($k, $this->fields)) {

                if ($k == $this->fields['pk']) {

                    // If it’s PK, construct where condition

                    $where = "`$k`=$v";

                } else {

                    // If not PK, construct update list

                    $uplist .= "`$k`='$v'".",";

                }

            }

        }

        // Trim comma on the right of update list

        $uplist = rtrim($uplist,',');

        // Construct SQL statement

        $sql = "UPDATE `{$this->table}` SET {$uplist} WHERE {$where}";
        if ($this->db->query($sql)) {

            // If succeed, return the count of affected rows

            if ($rows = mysql_affected_rows()) {

                // Has count of affected rows

                return $rows;

            } else {

                // No count of affected rows, hence no update operation

                return false;

            }

        } else {

            // If fail, return false

            return false;

        }


    }

    /**

     * Delete records

     * @access public

     * @param $pk mixed could be an int or an array

     * @return mixed If succeed, return the count of deleted records, if fail, return false

     */

    public function delete($pk){

        $where = 0; //condition string

        //Check if $pk is a single value or array, and construct where condition accordingly

        if (is_array($pk)) {

            // array

            $where = "`{$this->fields['pk']}` in (".implode(',', $pk).")";

        } else {

            // single value

            $where = "`{$this->fields['pk']}`=$pk";

        }
        // Construct SQL statement

        $sql = "DELETE FROM `{$this->table}` WHERE $where";

        if ($this->db->query($sql)) {

            // If succeed, return the count of affected rows

            if ($rows = mysql_affected_rows()) {

                // Has count of affected rows

                return $rows;

            } else {

                // No count of affected rows, hence no delete operation

                return false;

            }

        } else {

            // If fail, return false

            return false;

        }

    }

    /**

     * Get info based on PK

     * @param $pk int Primary Key

     * @return array an array of single record

     */

    public function selectByPk($pk){

        $sql = "select * from `{$this->table}` where `{$this->fields['pk']}`=$pk";

        return $this->db->getRow($sql);

    }

    /**

     * Get the count of all records

     *

     */

    public function total(){

        $sql = "select count(*) from {$this->table}";

        return $this->db->getOne($sql);

    }
    /**
     * Get info of pagination
     * @param $offset int offset value
     * @param $limit int number of records of each fetch
     * @param $where string where condition,default is empty
     */
    public function pageRows($offset, $limit,$where = ''){
        if (empty($where)){
            $sql = "select * from {$this->table} limit $offset, $limit";
        } else {
            $sql = "select * from {$this->table}  where $where limit $offset, $limit";
        }

        return $this->db->getAll($sql);
    }
}
```

Now we can create a User model class in our application folder, this is for our User table in the database. The code would look like this:

Hide Copy Code

```
<?php
// application/models/UserModel.class.php
class UserModel extends Model{

    public function getUsers(){
        $sql = "select * from $this->table";
        $users = $this->db->getAll($sql);
        return $users;
    }
}
```

And our application's backend indexController could be look like this:

Hide Shrink Copy Code

```
<?php

// application/controllers/admin/IndexController.class.php


class IndexController extends BaseController{

    public function mainAction(){

        include CURR_VIEW_PATH . "main.html";

        // Load Captcha class

        $this->loader->library("Captcha");

        $captcha = new Captcha;

        $captcha->hello();

        $userModel = new UserModel("user");

        $users = $userModel->getUsers();

    }

    public function indexAction(){

                       $userModel = new UserModel("user");

        $users = $userModel->getUsers();

        // Load View template

        include  CURR_VIEW_PATH . "index.html";

    }

    public function menuAction(){

        include CURR_VIEW_PATH . "menu.html";

    }

    public function dragAction(){

        include CURR_VIEW_PATH . "drag.html";

    }

    public function topAction(){

        include CURR_VIEW_PATH . "top.html";

    }

}
```

So far, our application backend's Index controller is working, it communicates with Model and passes result variable to the View templates, so it can be rendered in the browser.

This is a very brief introduction to a mini MVC framework, hope it clarifies some basic concepts in the MVC frameworks.

History

Feb 25, 2016 - initial version

License

Share

About the Author

Chris_Yu
Web Developer
Australia

Chris Yu is a full stack web developer, a Zend Certified PHP Engineer. His current interest focuses on front-end web component, as well as front-end & back-end separation development.

You may also be interested in...

A Solution Blueprint for DevOps

Solution Build Timer for VS 2005/2013/2015/2017/2019

MVC in PHP

Dynamic Treeview with Drag and Drop by Kendo

PHP MVC with .NET like controller

MicroMVC A Simple PHP Framework

Comments and Discussions

You must Sign In to use this message board.

Search Comments

First Prev Next

You ask for negative reactions

Member 14076627 3-Dec-18 21:00

Dear author,
Despite you started very well,
I know it's not nice to hear critique, especially negative but,
the way you teach/lecture and setup this tutorial is terrible from teaching point of view.

There are no examples of data going through your MVC (regardless of it's deprecated mysql). Also some code is not pointed to right filename. This is even more annoying for a student.

You should change the title: How to write bad tutorial for DIY MVC Framework.
Or best is to alter/improve or remove this piece of work.

Sign In · View Thread

Regarding code

Member 10409085 1-Oct-18 21:56

Here getcwd() would return the path hoti public folder according to given folder structure. So for this what should I do?

Thanks

Atul Kumar Tripathi

Sign In · View Thread

Have problem

Member 13982601 12-Sep-18 17:32

My erro is:

Hide Copy Code

```
Warning:
require_once(C:\xampp\htdocs\codeproject\application\controllers\home\IndexController.class.
php): failed to open stream: No such file or directory in
C:\xampp\htdocs\codeproject\framework\core\Framework.class.php on line 107
```

```
Fatal error: require_once(): Failed opening required
'C:\xampp\htdocs\codeproject\application\controllers\home\IndexController.class.php'
(include_path='C:\xampp\php\PEAR') in
C:\xampp\htdocs\codeproject\framework\core\Framework.class.php on line 107
```

Sign In · View Thread

Re: Have problem

Member 10409085 1-Oct-18 21:59

This is because of the "ROOT".
Here the ROOT is returning the address up to public folder.

Change ROOT value from getcwd to dirname(dirname (__FILE__)).

It should work and please tell me if it works.

Sign In · View Thread

add download link for source code

S.Shrestha 27-Jun-18 23:22

I love your article and thank you for posting it . Article was really helpful. It will be more helpful if you please provide the source code link or download link .

Sign In · View Thread

Please its a request

Kshitij Srivastava46 20-May-18 4:44

Can anyone give me the source code for this project...

Sign In · View Thread

PHP mvc Structure

Member 13594750 26-Dec-17 5:53

First of all Thanks for sharing PHP MVC Structure. I follow your code but face some error please help me
<a href="http://prntscr.com/hsinfv"></a>[<a href="http://prntscr.com/hsinfv" target="_blank" title="New Window">^</a>]

Sign In · View Thread

Modify .htaccess to allow .js .css and other file without redirection to index.php

prograc 16-Aug-17 16:52

i had to modify something at .htaccess file in order to allow other files

i have the following

```
<pre><IfModule mod_rewrite.c>
Options +FollowSymLinks
RewriteEngine on


RewriteCond ${REQUEST_URI} ^.+$
RewriteCond %{REQUEST_FILENAME} \.(gif|jpe?g|png|js|css|swf|php|ico|txt|pdf|xml)$ [OR]
RewriteCond %{REQUEST_FILENAME} -f [OR]
RewriteCond %{REQUEST_FILENAME} -d [OR]
RewriteCond %{REQUEST_FILENAME} -l

# Send request via index.php
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php/$1 [L]
</IfModule>
AddType text/css .css</pre>
```

Sign In · View Thread

AWESOME WORK!!!
Member 13256860 13-Jun-17 2:28

Keep up the good work, don't pay attention to those nerds wanting to complicate it just for nothing.
A system can be complex but made out of simple things.

They get angry because you made it simple and anyone can just learn to code it in 1 hour.

Sign In · View Thread

Re: AWESOME WORK!!!
Chris_Yu 25-Sep-17 3:41

Thanks a lot for your support 😊

Sign In · View Thread

Omg... Please remove this post immediatly.
vaso123 24-Feb-17 15:46

You just did every possible mistake what you can.

Directory structure: what if I want to use the same view from admin / frontend / partners / etc.. directory? Should I copy that view to every views subdirectory? Using /view/admin/userform.php from frontend is not so lucky. If you duplicate the file, project maintenance will be pain in the a...

Framework: the worst I've ever seen... statci methods? Really? How will you write unit tests for these methods? Always try to avoid static methods.

Hide Copy Code

```
$_REQUEST
```

... It's a joke. You should know that about a variable where is it come from.

Hide Copy Code

```
$GLOBALS['config']
```

, but why? Using globals like using eval, evil.

Autoloading:

Hide Copy Code

```
substr($classname, -10)
```

. What if I am working with microcontrollers, and I want to call one of my controller to microController? What if I want to autoload something what is not model neither controller? Where is namespace handling? The best way is to use proper namespaces.

Mysql. On stackoverflow with this class you will earn 15 downvotes in minutes. mysql_* fucntions are deprecated, and removed from php 7. Use mysqli_* or PDO instead. You have no protection for sql injection.

Why core model is not abstract?

Sorry to say this, but this whole code is a crap.

Sign In · View Thread

Re: Omg... Please remove this post immediatly.
Member 13256860 13-Jun-17 2:35

Why it does even need unit tests?

Core doesn't have to be abstract, why it has to be abstract?

Btw, what does stackoverflow have ANYTHING to do with this? The guy just posted a way to create a framework and that's it, better stay in stackoverflow and don't come out, your complain is a crap not the code.

Sign In · View Thread

Re: Omg... Please remove this post immediatly.
Chris_Yu 25-Sep-17 3:21

Thanks for your comments. This is just the first 101 introduction to PHP's MVC. It is literally an "Hello World" example to explain some concepts.

Sign In · View Thread

Do not use this code
Patrik Sokol 23-Feb-17 9:52

Feb 16 and still using "mysql_". Kidding me? Do not waste your time guys.

Sign In · View Thread

Re: Do not use this code
Member 13256860 13-Jun-17 2:37

Why not?

Sign In · View Thread

Never, ever use this as a base for a php MVC

Spacecake192 14-Nov-16 13:24

Come on bro this is absolutely terrible and insecure and giving php noobies a push in totally the wrong direction. Wheres the use of PDO, decent url routing etc. Anybody who uses this is leaving themselves and their data wide open. The use of mysql as a db has even been dropped by the people who maintain php, msqli would have been a better choice but even that is way outdated, PDO or nothing. I aint even looked to see if there is anything MD5 in there, (i daren't)

To anyone thinking of learning anything from this then just forget it (even the folder structure is overkill..), either code one up yourself from the decent examples out there or use a packaged framework. If you are still convinced that this is for you then go for it, but it will take you longer to put it right than it would to actually build a great skeleton MVC.

Sign In · View Thread

Re: Never, ever use this as a base for a php MVC

Chris_Yu 25-Sep-17 3:20

Thanks for your comments. Sure no one will use this example in Prod. This is just the first 101 introduction to PHP's MVC. It is literally an "Hello World" example to explain some concepts.

Sign In · View Thread

Source Files

kribo 28-Aug-16 14:46

Hi there,

Luv your article but would appriciate to be able to look at the source code.
Can you add the source code or post a git link so we can download it.
thanks

Kribo

Sign In · View Thread

Great Informations

ayhankurt 20-Aug-16 20:43

Thank you very much for this article. There is a lot of usefull information.

$charset = isset($config['charset'])? $config['charset'] : '3306';
I think must be charset here not port.

Thanks again.

Sign In · View Thread

thanks for the article, waiting for next...
Developer @ CCO 19-Jul-16 7:20

I'm learning PHP and MVC tech, this article is very good, I'm waiting the next article for the same topics, MVC is'nt easy, but the benefits pay the work, thanks

Sign In · View Thread

I am unable to call another view
Member 10103817 12-Jul-16 0:31

Hide Copy Code

```
I am trying to add another controller and the view but I don't know how to call it on the
browser.I have tried Controller/xxxx.php it is not working
```

Sign In · View Thread 5.00/5 (1 vote)

Refresh 1

General News Suggestion Question Bug Answer Joke Praise Rant Admin