

require 'mind'

Learn. Live. Love. Dev.

A Most Simple PHP MVC Beginners Tutorial

by [Neil Rosenstech](#)

04 Jan 2015, it was Sunday

A few months ago, a friend of mine started taking courses about web development. Like (almost) everyone else he began learning PHP and soon enough he was asked to develop a small handmade MVC application. So he went on the internet and started searching easily understandable tutorials, only to find out there was not that many good guides out there. Let's fix this.

Disclaimer: this tutorial is just my take on the subject. I do not guarantee that it will solve your problem. MVC is a design pattern really easy to grasp when you've been working with it, yet a bit odd when you come across the first time. The application I will create is probably perfectly imperfect, this was originally a quick afternoon draft from a guy that has been working with Ruby on Rails the last couple years, it goes without saying that PHP was remotely a memory. Anyways, if you find anything in this article that you think could threaten the future of the universe, please yell at me in the comments below.

Application Github repo: https://github.com/Raindal/php_mvc

Let's begin with the boring part.

MVC, what the f*?**

Right now I'm guessing you have at least a basic understanding of PHP. You've probably been building small websites or applications already, to get a general idea of the possibilities.

I suppose you've been starting with putting everything in one file and you quickly moved on to using "includes" and "requires" (better even `require_once`) just to make your code more readable, easier to follow for you or someone else. You see, that transition you made was for your code to be clearer, and therefore, maintainable. Maybe you even know about such things as Separation of Concerns and DRY (Do not Repeat Yourself). If you don't know about those I suggest you take a quick look at it cause you'll come across that stuff a lot more than you'd think if you plan on following the white rabbit.

Here are some useful resources on the matter, just get a vague idea, do not get stuck on those (read the introductions at least I guess) :

- [Separation of Concerns](#)
- [DRY](#)

Well, MVC is a design pattern that suggests you organize your code in a way concerns are properly separated, and yet interact with each other in a certain fashion.

MVC stands for Model View Controller as you may already know. That makes 3 types of concerns, I know this sounds a bit esoteric but I have faith you'll get it by reading on. Just to make things simpler, just imagine you have an application and in that application you have at least those 3 folders : models, views and controllers. Those folders contain specific files (no shit Sherlock). Now, those files are classified that way for a good reason: they have to carry on different tasks.

In the end: models fetch and mold data which is in turn sent to the view and therefore displayed. The controller is managing the process.

Building an example app

Let's get the basic stuff done: create a database **php_mvc** that contains a table **posts** with 3 columns **id** (INT PRIMARY auto_increment), **author** (VARCHAR 255, note that in a real life situation you could also put an index on this one as searching through posts with a specific author may happen quite often) and **content** (TEXT).

Let's move on to creating our first file: `index.php`.

The first thing it's going to do is making sure we have something to query the database any time later in the execution. As this is a pretty specific thing we're gonna put it in a specific file that we're going to require in our `index.php`.

Note that I'm working in WAMP. My project root is at `C:\wamp\www\php_mvc_blog`

index.php

```
1 <?php
2     require_once('connection.php');
3 ?>
```

Let's write the actual file.

connection.php

```
1 <?php
2     class Db {
3         private static $instance = NULL;
4
5         private function __construct() {}
6
7         private function __clone() {}
8
9         public static function getInstance() {
10             if (!isset(self::$instance)) {
11                 $pdo_options[PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
12                 self::$instance = new PDO('mysql:host=localhost;dbname=php_mvc', 'root', '', $pdo_options);
13             }
14             return self::$instance;
15         }
16     }
17 ?>
```

You may have to change the 12th line to match your database user/password. Here my user is root and I do not have a password.

As you may have noticed we're just creating a singleton class. This allows us to return an instance of the connection and always the same as we only need one to execute all our queries. As the class is a singleton, we make `__construct()` and `__clone()` private so that no one can call `new Db()`. It has an `$instance` variable that retains the connection object (PDO here). In the end, we have access to our "connection object" through `Db::getInstance()`.

Note that singletons are not the only way to handle that, it's just been a common way for years but this may not be the best approach depending on your needs: [see this post on StackOverflow for instance](#).

Anyways, we can go on with our index file.

index.php

```
1 <?php
2     require_once('connection.php');
3
4     if (isset($_GET['controller']) && isset($_GET['action'])) {
5         $controller = $_GET['controller'];
6         $action      = $_GET['action'];
7     } else {
8         $controller = 'pages';
9         $action      = 'home';
10    }
11
12    require_once('views/layout.php');
13 ?>
```

Here is the whole file. Note that this index.php file is going to receive all the requests. This means that every link would have to point to `/?x=y` or `/index.php?x=y`.

The if statement is gonna check whether we have the parameters controller and action set and store them in variables. If we do not have such parameters we just make **pages** the default controller and **home** the default action. You may not know where we're going with this but you should get the idea by reading on. Every request when hitting our index file is going to be routed to a controller (just a file defining a class) and an action in that controller (just a method).

Finally we require the only part of our application that does not (theoretically) change: the layout.

Let's create this file.

views/layout.php

```
1 <DOCTYPE html>
2 <html>
3   <head>
4   </head>
5   <body>
6     <header>
7       <a href='/php_mvc_blog'>Home</a>
8     </header>
9
10    <?php require_once('routes.php'); ?>
11
12    <footer>
13      Copyright
14    </footer>
15  </body>
16 </html>
```

As you can see, we put it in a new *views/* folder because it is something displayed on the users' screen, in other words it is rendered. It contains only our header with the different links you could find in a basic menu and a footer. Note that while I'm working on WAMP for this guide the root of my application is not / but /php_mvc_blog (the name of my folder under www/ in WAMP).

In the middle we require another file: routes.php. The only part we still need is the main area of our page. We can determine what view we need to put there depending on our previously set *\$controller* and *\$action* variables. The routes.php file is gonna take care of that.

This file is not in the views folder but at the root of our application.

routes.php

```
1 <?php
2 function call($controller, $action) {
3   // require the file that matches the controller name
4   require_once('controllers/' . $controller . '_controller.php');
5
6   // create a new instance of the needed controller
7   switch($controller) {
8     case 'pages':
9       $controller = new PagesController();
10      break;
11    }
12
13   // call the action
14   $controller->{ $action }();
15 }
16
17 // just a list of the controllers we have and their actions
18 // we consider those "allowed" values
19 $controllers = array('pages' => ['home', 'error']);
20
21 // check that the requested controller and action are both allowed
22 // if someone tries to access something else he will be redirected to the error action of the pages controller
23 if (array_key_exists($controller, $controllers)) {
24   if (in_array($action, $controllers[$controller])) {
25     call($controller, $action);
26   } else {
27     call('pages', 'error');
28   }
29 } else {
30   call('pages', 'error');
31 }
32 ?>
```

Ok so we want our routes.php to output the html that was requested one way or another. To fetch the right view (file containing the html we need) we have 2 things: a controller name and an action name.

We can write a function *call* that will take those 2 arguments and call the action of the controller as done in the previous code sample.

We're doing great. Let's continue and write our first controller for the call function to find the file.

controllers/pages_controller.php

```
1 <?php
2 class PagesController {
3     public function home() {
4         $first_name = 'Jon';
5         $last_name = 'Snow';
6         require_once('views/pages/home.php');
7     }
8
9     public function error() {
10        require_once('views/pages/error.php');
11    }
12 }
13 ?>
```

The class is rather straightforward. We have 2 public functions as expected: `home()` and `error()`. As you can see, the first one is also defining some variables. We're doing this so we can later output their values in the view and not clutter our view with variables definition and other computation not related to anything visual. Separation of concerns.

To keep things simple, we usually name the view after the action name and we store it under the controller name.

Let's create both our views.

views/pages/home.php

```
1 <p>Hello there <?php echo $first_name . ' ' . $last_name; ?>!<p>
2
3 <p>You successfully landed on the home page. Congrats!</p>
```

Use those previously defined `$first_name` and `$last_name` where you see fit.

views/pages/error.php

```
1 <p>Oops, this is the error page.</p>
2
3 <p>Looks like something went wrong.</p>
```

Fantastic! We have a working sample of our MVC app. Navigate to *localhost/php_mvc_blog* and you should see the following:

```
Home
Hello there Jon Snow!
You successfully landed on the home page. Congrats!
Copyright
```

Improve the app with models and some data management

What we want now is to be able to query our database in a clean way and display the results. Say we want to be able to fetch a list of posts and display those, and same thing for one particular post.

Let's modify our *routes.php* file to reflect that behaviour.

Here is the final file:

routes.php

```
1 <?php
2 function call($controller, $action) {
3     require_once('controllers/' . $controller . '_controller.php');
4
5     switch($controller) {
6         case 'pages':
7             $controller = new PagesController();
8             break;
9         case 'posts':
10            // we need the model to query the database later in the controller
11            require_once('models/post.php');
12            $controller = new PostsController();
```

```

13     break;
14 }
15
16 $controller->{ $action }();
17 }
18
19 // we're adding an entry for the new controller and its actions
20 $controllers = array('pages' => ['home', 'error'],
21                     'posts' => ['index', 'show']);
22
23 if (array_key_exists($controller, $controllers)) {
24     if (in_array($action, $controllers[$controller])) {
25         call($controller, $action);
26     } else {
27         call('pages', 'error');
28     }
29 } else {
30     call('pages', 'error');
31 }
32 ?>

```

Just pay attention to where I add a comment, those parts where modified.
We can create the posts controller now and we will finish with the model.

controllers/posts_controller.php

```

1 <?php
2 class PostsController {
3     public function index() {
4         // we store all the posts in a variable
5         $posts = Post::all();
6         require_once('views/posts/index.php');
7     }
8
9     public function show() {
10        // we expect a url of form ?controller=posts&action=show&id=x
11        // without an id we just redirect to the error page as we need the post id to find it in the database
12        if (!isset($_GET['id']))
13            return call('pages', 'error');
14
15        // we use the given id to get the right post
16        $post = Post::find($_GET['id']);
17        require_once('views/posts/show.php');
18    }
19 }
20 ?>

```

As you may have noticed, the pages controller is managing static pages of our application whereas the posts controller is managing what we call a resource.

The resource here is a post. A resource can usually be listed (index action), showed (show action), created, updated and destroyed. We will keep the focus on the first two as building a fully functional app is not the point of this article.

`Post::all()` and `Post::find()` refer to the model `Post` that has not yet been written. One of the best piece of advice I was ever given was to write the final code as I want it to be. This way I'm sure I'm writing the most beautiful code I can come up with and then I make sure it works.

So here, when I fetch the posts I want it to look like `$posts = Post::all();` as this is clear and simple.

Now we can move on to writing the model.

models/post.php

```

1 <?php
2 class Post {
3     // we define 3 attributes
4     // they are public so that we can access them using $post->author directly
5     public $id;
6     public $author;
7     public $content;
8
9     public function __construct($id, $author, $content) {

```

```

10     $this->id      = $id;
11     $this->author  = $author;
12     $this->content = $content;
13 }
14
15 public static function all() {
16     $list = [];
17     $db = Db::getInstance();
18     $req = $db->query('SELECT * FROM posts');
19
20     // we create a list of Post objects from the database results
21     foreach($req->fetchAll() as $post) {
22         $list[] = new Post($post['id'], $post['author'], $post['content']);
23     }
24
25     return $list;
26 }
27
28 public static function find($id) {
29     $db = Db::getInstance();
30     // we make sure $id is an integer
31     $id = intval($id);
32     $req = $db->prepare('SELECT * FROM posts WHERE id = :id');
33     // the query was prepared, now we replace :id with our actual $id value
34     $req->execute(array('id' => $id));
35     $post = $req->fetch();
36
37     return new Post($post['id'], $post['author'], $post['content']);
38 }
39 }
40 ?>

```

Our model is a class with 3 attributes.

It has a constructor so we can call `$post = new Post('Neil', 'A Most Simple PHP MVC Beginners Tutorial')` if need be. I think this is pretty much straightforward but let me know in the comments below if something needs more explanation.

Let's modify our layout to add a link to the posts in the header.

Here is the final file.

views/layout.php

```

1 <DOCTYPE html>
2 <html>
3   <head>
4   </head>
5   <body>
6     <header>
7       <a href='/php_mvc_blog'>Home</a>
8       <a href='?controller=posts&action=index'>Posts</a>
9     </header>
10
11     <?php require_once('routes.php'); ?>
12
13     <footer>
14       Copyright
15     </footer>
16   </body>
17 </html>

```

Ok now final step we have to create the views we require in the posts controller.

views/posts/index.php

```

1 <p>Here is a list of all posts:</p>
2
3 <?php foreach($posts as $post) { ?>
4   <p>
5     <?php echo $post->author; ?>
6     <a href='?controller=posts&action=show&id=<?php echo $post->id; ?>'>See content</a>
7   </p>
8 <?php } ?>

```

And

views/posts/show.php

```
1 <p>This is the requested post:</p>
2
3 <p><?php echo $post->author; ?></p>
4 <p><?php echo $post->content; ?></p>
```

We need to have some data in the database so add some manually as we do not have a feature for this yet. I'm using phpmyadmin for this matter.

I created 2 posts :

- one with author “Jon Snow” and content “The Wall will fall.”
- the other with author “Khal Drogo” and content “I may be dead but I’m still the most awesome character.” (and you shall not disagree on that one)

Now go ahead and test everything in your browser.

What you’ve seen here is a way of separating the concerns in an MVC style. Note that a lot of stuff is open to interpretation. This way of seeing things is mostly inspired by what I’m used to do with Rails.

If you want to go further down the road you should check out some framework like Symfony for PHP, Django for Python or Rails for Ruby.

That’s it folks! Hope it helps. Until next time.

by [Neil Rosenstech](#)

Share this post

[Twitter](#) [Facebook](#) [Google+](#)

[« Memoization, speed up your javascript performance /](#) [Home](#)

145 Comments

require 'mind'

Login

Recommend 26

Share

Sort by Best




Join the discussion...

LOG IN WITH


OR SIGN UP WITH DISQUS ?

Name

- 

Jojo • 3 months ago

Thanks you so much for this awesome article about MVC. Actually I have been looking for something like this. It so clean and perfect (Simple comments with clean explanation). I will use your code in my projects. Thanks again. Now i am going to check the php version, where i can use class objects.. ecc.. Anyhow thanks you so much.

14 ^ | v • Reply • Share ›
- 

Dimitris Selimidis • a year ago

Which .htaccess will best match this MVC ? Utilizing query string as well (like pagination)

5 ^ | v • Reply • Share ›



Pratibha Mishra • 2 years ago

Amazing work. I just went step by step and wow its working.
Its really great article if anybody has to start as a beginner !!
Thank you

2 ^ | v • Reply • Share ›



Vikas kumar jain → Pratibha Mishra • a year ago

who's this pratiksha mishra

^ | v • Reply • Share ›



Sheikh Heera • 3 years ago

Actually, Model–view–controller (MVC) is a software architectural pattern (<http://en.wikipedia.org/wik...> which tells how to utilize/organize resources effectively, while the design pattern is a general reusable solution to a commonly occurring problem within a given context in software design and it tells how to write code effectively for better maintainability and re-usability. Both are different things and the main motivation behind the MVC is SoC (Separation of Concerns) design principle and hence most of the web frameworks are not 100% MVC (According to some other experts) but I think it's fine if they follow the SoC design principle.

To be honest, in your example, I can see you didn't follow the SoC design principle and this example is not appropriate for this era of web engineering. Alos, Symfony follows the HMVC (Hierarchical Model View Controller) which is known as component based/modular but still follows the SoC principle.

You should Google the web for MVC examples/tutorials and Check the Laravel framework if you didn't yet. Btw, your article could lead others to wrong direction.

3 ^ | v • Reply • Share ›



require mind Author → Sheikh Heera • 3 years ago

With all due respect, I think deep reasoning about terms like design/architectural pattern is a terrible rationale. This article is aimed at people having no idea at all of what MVC is or that are struggling with it. Expanding on what you're saying would be like explaining rocket science to a first year engineer student.

Actually, I did Google for PHP and MVC stuff, only to find out most articles do a great job at spreading complicated terms with complicated examples instead of giving basic explanations on basic stuff.

Ultimately, if people reading my article end up with a better general idea of the concept then it will be mission accomplished for me and I really don't care if they are wrong using the term "design pattern" instead of "architectural pattern".

8 ^ | v • Reply • Share ›



azwaldo → require mind • 2 years ago

Clearly written; good flow, from one bit to the next.

Commenting as one who had *"no idea at all of what MVC is"*, and as someone who was definitely *struggling with it* after several articles/introductions, I now have a better understanding of SoC after building a simple application. That was only possible for me by using the example in this tutorial.

Much appreciation, here.

2 ^ | v • Reply • Share ›



require mind Author → azwaldo • 2 years ago

Awesome! This is why I keep writing stuff and comments like yours are the best rewards. I'm glad it helped ;)

1 ^ | v • Reply • Share ›



Azad Hind Bhagu → require mind • 2 years ago

Heartiest thanks to you friend. Your effort is something I needed very much. Being a Scholar is a different thing. Being a teacher is another. People sometimes get into a habit of looking at the glass from top (and thus end up yelling the empty part)...;) ...=D..... Thanks for making the world even better. :) It's far better than it has been explained on Symfony's website.

^ | v • Reply • Share ›



require mind Author → Azad Hind Bhagu • 2 years ago

Thanks! I didn't have the chance to read much about symfony but from my understanding it has a slightly different mechanism due to its use of components. From what I saw it's just some sort of "improved" MVC where concerns are further split into more entities.

^ | v • Reply • Share ›



Brian Moeller → require mind • 2 years ago

I thought the article was clear, simple and easy to follow. Someone who has no knowledge of MVC will get a great start with this post. The over analysis on the pure definition of MVC is over board by some of the posters. If you're working for Google and need to maintain the highest levels of standards in order to provide the most effective code created by man, then have at it. Let the other 99% of the web developers out there putting together sites use the MVC (or any other design pattern) as a loose framework for building their next web application.

There is always room for improvement, but I could give two \$hits if I didn't have the purest pattern. I'm making websites, not space ships.

^ | v • Reply • Share ›



jylipaa → Brian Moeller • 2 years ago

Yet again, article was clear and very good tutorial for people to start with. The code works very well and there is nothing wrong with that.

But calling things with wrong names will only cause more and more confusion. That's the point here.

2 ^ | v • Reply • Share ›



Chess_is_great → jylipaa • 2 years ago

Is Laravel following MVC/SOC as it should be? Looking at Laravel controller, it also fetch data from model and its controller pass data to view.

1 ^ | v • Reply • Share ›



jylipaa → Chess_is_great • 2 years ago

As far as I know, Laravel developers doesn't call Laravel a MVC framework.

It looks more like MVP (Model-View-Presenter) to me. In MVP, the Presenter is designed to pull data from model and push it to views.

^ | v • Reply • Share ›



Chess_is_great → jylipaa • 2 years ago

I see. So i guess the example here used MVP. Which do you think is better? MVC or MVP?

^ | v • Reply • Share ›



jylipaa → Chess_is_great • 2 years ago

I think it's up to goals and desires. MVP cuts some corners, but will produce easily maintained code. Sure the "controllers" (Presenter) will have more than one responsibility. I am a nihilist, i try to enforce MVC just for the curiosity when I don't have deadline pressures on me. Otherwise, I use MVP :)

^ | v • Reply • Share ›



lynismo → jylipaa • a year ago

jylipaa, you are absolutely correct in your statements above; no it is not MVC. I think that people are understandably dismissive about the strict definition of the term, but it is indeed important to understand that this is not MVC, and why it matters to them (or at least it should).

I like the MVP pattern myself, for the very reasons you mentioned -- code maintenance is much easier, which is a significant attribute to have -- but it should also be noted that MVP makes it a good bit more difficult to make truly modular code. Which may seem trivial to newer programmers, but once they dig in a bit and write something even remotely complex, they'll begin to understand how a true MVC pattern will help them make reusable code.

A good, and common, example of that is when trying to use AJAX to pull data from models. If you're using the "Presenter" to pull data from the model and push it to a view, and the view is just a dummy template that is tightly coupled with the Presenter, you aren't fetching just the data that AJAX requested, you're also fetching an entire web page. I can't imagine many scenarios where that is desirable...that's just one of many example of why not to consider a view nothing more than a template file.

Unfortunately, a lot of popular software/frameworks out there label themselves as MVC when they are not, which contributes to all the confusion.

1 ^ | v • Reply • Share ›



Robert Talada → Sheikh Heera • 2 days ago

There are a lot of mechanics/plumbers/engineers that know how to accomplish a task without necessarily knowing exactly what they are doing is called. It's the difference between being book-smart or street-smart. If everyone would just design appropriate applications for the task at hand instead of applying generic, bloated, one-size-fits-all methodologies to every task at hand, applications would perform much better in general.

^ | v • Reply • Share ›



mohsin → Sheikh Heera • 2 years ago

nice dud

^ | v • Reply • Share ›



Rory McCaffrey • 2 years ago

Good stuff. I've been looking for tutorials, couldn't find any that i could get my head around and this is perfect to get me started. thanks man :)

1 ^ | v • Reply • Share ›



Terere Ruso → Rory McCaffrey • a year ago

Great job, does anyone made a login to add to this example?

^ | v • Reply • Share ›



Garth • 2 years ago

Excellent for an MVC beginner - thanks!

1 ^ | v • Reply • Share ›



fan • 2 years ago

thank's dude,,
you genius.

1 ^ | v • Reply • Share ›




Jani • 2 years ago

In SoC point of view, your Controllers are here doing many tasks right now - e.g. they are fetching data from model and then they are also responsible for rendering views right now! So, Controllers here are getting fat and not separated by

then they are also responsible for rendering views right now. So, controllers here are getting fat and not separated by concern.

As another note, View in MVC isn't just a dummy template, it's a class of its own, responsible of fetching data from Model (which MIGHT have been altered by Controller, not necessarily), then decide - based on that data - how to display it. E.g. using templates.

While this code works and probably is a fast and efficient way to develop things, it's not MVC. It's not bad code, but it's not MVC either.

1   • Reply • Share ›



Tushar Sharma → Jani • a year ago

I think everyone implements MVC a little differently. I was trying to learn CakePHP, even signed up for an online course. Sadly, the instructor didn't explain MVC well, but in his implementations Model and View never communicated directly - the controller did all the talking. I guess it makes it more like layered architecture. On Wikipedia, I see something else, with model and layer talking to each other, in addition to the controller.

BTW, I'd like to see how View classes can be implemented, and what benefits they provide.

  • Reply • Share ›



sssssss → Jani • 2 years ago

right

  • Reply • Share ›



require mind Author → Jani • 2 years ago

I'd say you're half right on that one, let me explain why. The controllers are not directly fetching data, the models are. Also, just because the directive for including the view stands right there in the controller does not mean the concerns are not separated.

At the end of the day, the controller is delegating to some model (as it should, this could be improved but the basic operations are there) and rendering the view, not through a separate class as you mentioned but rather directly because for this example I did not want to clutter my code with useless things. A framework like Rails does a lot of transformation under the hood that I do not need to do here so using a class for that seemed overkill.

Sure there are a number of ways through which you could improve this, like extracting domain specific objects and having a controller doing 100% delegation.. but once again this lies beyond the scope of this article.

  • Reply • Share ›



Blu Tiger → require mind • a year ago

Can I just say..."head exploded?" Lol.

1   • Reply • Share ›



Jani → require mind • 2 years ago

In this example, your controller is fetching the data through a model and then passes it to view. See, it has now two responsibilities which isn't SoC and definitely not mvc. Controller alters the state of the model, view then renders itself based on model. Not based on what controller passes from the model, it renders itself by the state of the model layer. I don't say your example is bad, it's just not MVC.

  • Reply • Share ›



require mind Author → Jani • 2 years ago

"it renders itself" haha, yes, by the magician's wand ;) It is SoC and in that case the controller's responsibility is to organize data and dispatch "actionsactions" to be taken to whatever other structure responsible for it. Saying fetching data through the model and passing it to the view are two responsibilities is like saying rendering the title and rendering the subtitle are two different things... damn, concerns are not separated properly in the view now. By the way, this is all mental

masturbation, in the end, do you write good software/apps or not is what matters.

1 ^ | v • Reply • Share ›



jylipaa → require mind • 2 years ago

So, if someone disagrees with you, you start removing comments. Cool, man :)

Here it is, one more time:

"Well, Title and subtitle are both view concerns, so it really isn't the same thing. And you know, you can call view class' output methods just like you call controller's action method - outside the MVC triad.

Controller alters the model, view then renders itself (after your routes.php calls controller's action method) based on whatever model contains. Or by using Observer pattern where Model tells View it has changed. And tadah, there you have a mvc triad with concerns way better separated.

Don't get offended when people correct some issues rather than just taps your back and says that you rule. And yet again, i am not saying your code is bad (because it works and does what it promises), it just is not MVC."

1 ^ | v • Reply • Share ›



require mind Author → jylipaa • 2 years ago

I deleted it because I just don't have time to play ball with you on this (no sarcasm here) these days and I always like to answer comments, that I agree or not with. You commented, I answered, if you comment again it's never ending. We hear you, let's keep it there and not be childish.

Thanks for reading and commenting ;)

^ | v • Reply • Share ›



Aleksandar → Jani • a year ago

That concept use Joomla and after 5 years in Joomla, found RoR, Yii, and... and after all leave Joomla and hate when need to develop some component for Joomla (clients task). And, do not complicate. I don't care about fat controller, something at the end will be fat :), or to many granularity in structure is it ok? regards

^ | v • Reply • Share ›



Steve Ralston • 13 days ago

This was wonderfully useful to me, thank you!

^ | v • Reply • Share ›



Guilherme Machado • 20 days ago

hey author, how can we make a login form with this example? I have a page with inputs for user and password, and in pages controller I did the basic like this article, and a model to check if user exist on database.. my doubt is:

if user exists, where can I redirect to the panel, on model or controller? I can't figure out. Will my form have an action attribute? because I need to create 2 functions on controller, the admin() - page with form - and panel() - the panel itself -. how can I redirect the user either the answer is true or false? thanks!

^ | v • Reply • Share ›



My NIQQA • a month ago

How does the index.php isset know if i lets say click another link? isnt the file executed from top to bottom? How does it make it way back to the top?

^ | v • Reply • Share ›



SiamIT Kittichai • a month ago

ขอบคุณมากครับ

Wow Thank You for framework

^ | v • Reply • Share ›



Brad Chellingworth • 2 months ago

Awesome tutorial. Thanks

^ | v • Reply • Share ›



Theo Dores • 2 months ago

Hai, a simple question.

How can I show a view in sub folder?

Lets say, I want to display a file in views/posts/somefolder/simplecontent.php

Thank you in advance.

^ | v • Reply • Share ›



MoronVV • 2 months ago

This is a best pure php MVC solution! I was looking for it soooo long!

^ | v • Reply • Share ›



Andrés Rodríguez • 3 months ago

Awesome! I didn't know how to implement this pattern but this post made me put all my ideas together. Thank you very much, greetings from Mexico.

^ | v • Reply • Share ›



Ozan Kurt • 3 months ago

PERFECT!

^ | v • Reply • Share ›



Bardhan Sumi • 4 months ago

If you please add a project with registration and login along with few pages, will create better idea.. thanks..

^ | v • Reply • Share ›



Justice Igwe • 4 months ago

God bless the author of this, i don't care what pattern or so people follow. the truth is Before all those came out, they started from somewhere and got there, so this admin has done a great job. thanks . i now get the full navigation on how to ride the MVC horse because of you. Thanks once again. God bless

^ | v • Reply • Share ›



leo • 4 months ago

hello i want to asking about error :

Parse error: parse error in C:\xampp\htdocs\php_mvc_blog\routes.php on line 20

^ | v • Reply • Share ›



hetal shah • 4 months ago

Explained So nicely but you didn't specify which file should be in which folder

^ | v • Reply • Share ›



ion • 5 months ago

There is nothing "simpler" in this code...

^ | v • Reply • Share ›



Cha OS • 5 months ago

Great Job, helped a lot :-)

^ | v • Reply • Share ›



lolka_bolka • 5 months ago

Database as a singleton is a really bad idea. What if I need to connect multiple databases? Static methods are also not good design, always try to avoid them. I know you want to point the MVC for really beginners, but don't suggest to them these bad habits,

^ | v • Reply • Share ›



Goujon • 5 months ago

Really great article but like some people noticed, every MVC starting with right .htaccess and specifically ModRewrite rules in it. Site's "single entry" concept plays well here. No matter what address the user is trying to access the site through they will be always redirected to one and only index.php which then analyze the syntax of address and redirects (routes) to specific controller to take specific actions accordingly. So maybe it is worth expanding the article on that matter?!

^ | v • Reply • Share ›

[Load more comments](#)

ALSO ON REQUIRE 'MIND'

Linux command line Tips: Become a master

11 comments • 4 years ago •



Matt Rose — && and ; are not equivalent. try # false ; echo "done"# false && echo "done"

Differences between has_one and belongs_to in Ruby on Rails

19 comments • 4 years ago •



Dave Aronson — I keep this straight by thinking of physical possessions with a name on them. When I buy, for instance, a book I might put my name inside the cover ...

[Subscribe!](#)

Follow us on [Twitter](#) [GitHub](#) [Google+](#)

All content copyright [requireMind](#) © 2015 • All rights reserved • Published with [Octopress](#)

Deploying a Rails App on Your Own Server - The Ultimate Guide

5 comments • 3 years ago •



asksammy — Thanks for the post. I have redis on a separate server. Should sidekiq be deployed on same or different server as the ruby app?

Ruby on Rails 4 and Batman.js - Another Getting Started Tutorial

15 comments • 4 years ago •



require mind — Nice ! And to answer your previous question I could not really recommend any other tutorial for two reasons : I've tried Batman and Angular so far but I ...