The Blog!

Liu Xinan

January 5, 2015

Some background

This blog has not changed much since mid-submission. Just some fine-tunings and bug-fixes.

Initially I made a even more naive version of this blog. The structure is not clear enough so I decided to rewrite it in a MVC style.
Having "do not start from scratch" and "do not reinvent the wheels" in mind, initially I was seeking for some PHP MVC framework, and I tried CakePHP and Laravel. Then I found that using such frameworks makes the task too easy to complete. I guess that is not the intention of this assignment although it says do not start from scratch. For some simple ones, I don't like the way they are being implemented. Therefore I started to write my own MVC framework. This is what a hacker do, isn't it? If you don't like something, make one yourself :)

Honestly speaking, I didn't learn MVC in a systematic way, so my understanding of MVC might be naive. But I think it is always good to try out something new. (Likewise, this is also my first time using LaTeX.) Although I know that it is not good to reinvent the wheels, I still did that. Every piece of code, except for css and the editor plugin, is written by myself. But of course, I did read some others' code and learnt some nice tricks from them.

1 Features

It is just a simple blog site. People can sign up, login and post articles. All articles are viewable at the home page, without the need to sign in.

1.1 Home page

At the home page, post abstracts are listed according to their publish date in descending order. Abstracts are auto-generated by extracting the first 30 words from a post. Below each title, there is some metadata. Clicking on the name of the author leads to a page showing all posts from that author. Click "Read more" shows up the entire post.

1.2 Sign up page

Sign up requires a first name, last name, a alphanumerical username that begins with a letter, an email address, and a password that contain at least 8 characters, consisting of uppercase and lowercase letters as well as numbers.

1.3 Login page

Users can login with either their username or email address, a "remember me" option is provided. The remember me cookie expire in 30 days, but each login via cookie will renew that cookie.

1.4 Profile page

The profile page shows some basic information about the user, such as username, email, gender, and how long they have been registered. There is also entrances to write a new post, view one's own posts, as well as to change password or edit profile.

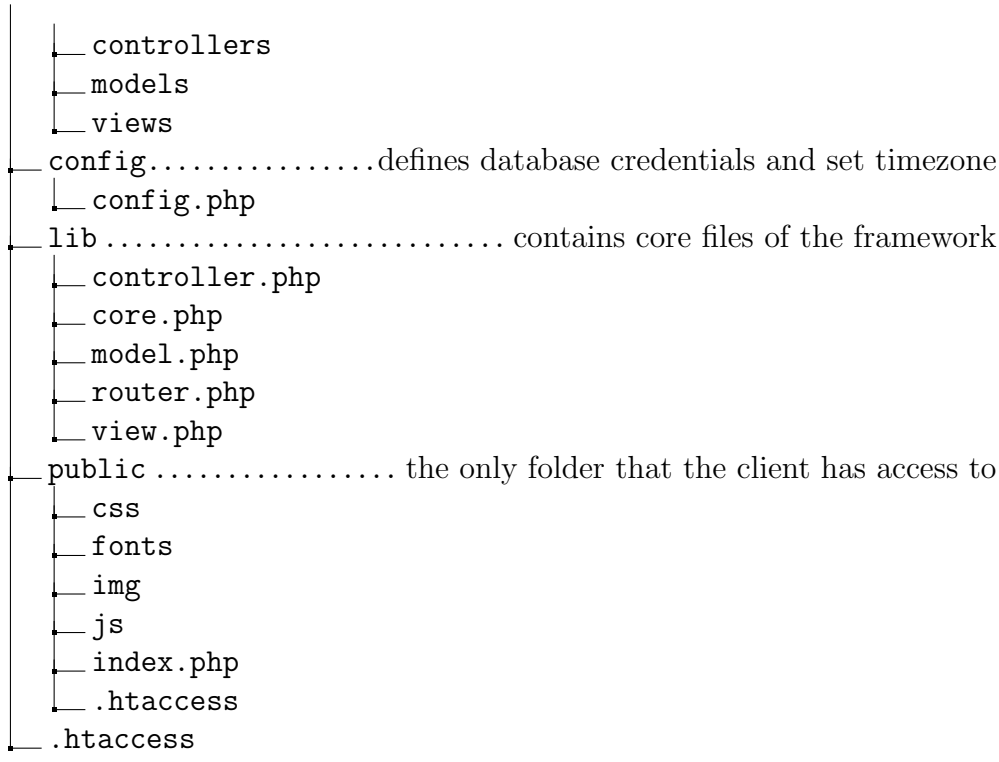
1.5 Post features

Users can view, edit or delete his/her posts. The edit and delete button only appears on hover. I used a WYSIWYG editor js plugin called "tinyMCE" for the content of the post. It supports a wide range of formatting styles. But for neatness and security reasons I disabled some of them. But it still enough to format an article.

2 Technical Details

2.1 The directory tree

```
/
└── app ..... the main app, contains most of the code
```



2.2 How the MVC framework works

In this subsection I will briefly explain how this MVC framework works. The two `.htaccess` files are very important. I use them to enable `mod_rewrite` and redirect the requests to `/public/index.php`. This is how a request is processed:

1. A URL of the form `hostaddress/controller/action/param1/param2/...` is requested. The `/public/index.php` gets loaded.

2. `index.php` loads the config and core files of the framework, and instantiates a `Router` object, and then ask that router to route request.

3. When a `router` is constructed, it parses the request URI and split it into 3 different parts: `controller`, `action`, and an array of `parameters`.

4. Then the `routeRequest()` method gets called, it start to look for the `controller`, pass the `parameters` to the `controller`, then execute the `action` under that `controller`. If a `controller` is not specified or the `controller` cannot be found, it will redirect to the home page. If a `controller` is found but the `action` is not found, by default it will execute the `index()` action.

5. The `controller` does what it needs to do with the `models`, creates a `View` object and binds some necessary variables to it, then invokes the `render()` method of the `view` to render the view.

2.3 Security

These are some security aspects that I've looked into:

- Prepared statements are used to prevent SQL injection.
- Every input are escaped before writing into the database, to prevent script injection.
- Passwords are stored as a hash, using the build-in `password_hash()` function (PHP 5.5+). Therefore even if the database is leaked, the passwords won't be exposed.
- The main codes are all outside the `public` folder, which the user have no access to.
- "Remember me" cookies are also hashes, which are generated from the hashed password. Therefore changing the password can invalidate all existing cookies.
- User have to be the owner of a post to be able to edit and delete the post.
- Disabled server directory listing

2.4 Miscellaneous

- I used `Bootstrap` to do some simple styling
- Adhere to the DRY principle, View utilises some basic templates and partials that I've made to reduce duplicate codes. It loads the application layout, then substitute title and content.
- URLs are clean and pretty. And have the potential to be SEO friendly.
- Defined `__autoload()` to load the model and controller classes automatically.
- The `Model` provides an abstraction layer. Interaction with database can be done in a OOP style.

- Front-end codes are completely separated from back-end codes.
- Make use of late static binding to avoiding creating duplicate objects.
- Used `utf8mb4` collation so that it supports `emoji`.
- Supports html formatting in post content rather than plain text only.
- Did not use CDN version of `Bootstrap` and `tinyMCE` because of the special Internet environment in China. (too slow for local testing)

3 What's lacking

This MVC blog is way far from perfect, there is still a lot to be done. Here is some TODOs:

- Implement a comment and 'Like' system.
- Sharing feature. (Through Facebook, Twitter, email, etc.)
- Better styling and responsive design.
- Support image and file upload.
- Support embed media.
- Reset password feature.
- Use POST instead of GET when deleting posts.
- Brute force prevention, restrict login after 5 failed attempts.

So far I have only done the "Better styling and responsive design" because of lack of time due to CS1010R and some other projects. I seek your understanding.

4 Ending

I know this article is exceeding the 3-page limit. But since there is so many blanks and the long directory tree, please bear with me.

Thanks for reading. I feel quite accomplished being able to create such a blog from scratch in one week, without any prior knowledge about web programming.