My Blog System Final Report

User Manual

This blog system provides a very basic template / prototype as a Content Management System (CMS). It is meant for study or academic purpose rather than any commercial or commercial-related purpose.

Special Notice: This project is under GNU General Public License (GPL) 3.0, which means you can redistribute it and/or modify it or any part of it. However, all such derivative work MUST be under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Below, a few key features of this system will be highlighted:

- **User authentication**

This system provides a very simple login page to allow certain pages to be visited by authorized users ONLY.

This distinguishes the users of this system into two classes: visitors and writers. For visitors can only visit the homepage and navigate to the page of any published articles; for writers, they can go to the control panel and write new articles and/or edit or delete existing articles as they like.

- **Control panel**

This system provides a simple control panel page for writers to manage the articles being posted. This is a one-stop control center of this system. All administration stuff can be finished here. Basically, they can do three kinds of operations here:

1) Write a new post;
2) Edit or modify an existing post;
3) Delete an existing post.

These three functions will be explained in details one by one later.

➢ *Write a new post*

When a writer wants to create a new article, he/she will be redirected to a new page, where the title and content of that article should be filled in. After the form is completed properly and submitted, the database will handle this request and save all the information for that article (author, title, content, and time of posted/lastly modified).

After all of these has been finished, the user will be redirected to the static link of this new article so that he can review his/her work.

➢ *Edit or modify an existing article*

In the control panel, the writer can select one article to edit. After selecting one of them, he /she will be redirected to a new page, where he can edit it based on the previous edition. After the modification is finished, the writer can submit the changes.

Then, the information both saved at the database and static link will be updated. In the end, the writer will be redirected to the static link of that article.

➢ *Delete an existing post*

Any of the existing posts can be deleted by any one of the writers in the current version (which may be changed in the future version of this system).

Writers should take note that a window is expected to pop up before the operation of delete is done so as to avoid mistakes.

- **Static link generation**

Each existing post has a specific and unique Uniform Resource Locator (URL) link. In this way, anyone can save the links of their favorite articles and visit them the next time. One of the major benefits of this design is to decrease the number of times of interaction with the database server. Less frequency of performing queries to the database means easing the burden of the database server.

As a reader of this manual, you are very first batch of users to this system. This means you can expect a lot of new features to be added into this system in the very near future.

A few potential features in the next version are listed as follows:

- **New user registration**

Currently, all users are by invitation only and the information will be added to the database by the system administrator manually. This method is quite inconvenient and unsafe.

In the next version, new users can register by themselves and become a member of this system.

- **More roles**

There will be 3 kinds of roles: administer, writer and visitor. Different from now, writers can only edit/delete articles posted by themselves. However, administers can edit/delete all.

- **Comment system**

No matter which your role, everyone can comment below every article. Nonetheless, they can be deleted by administers.

A few advices would also be provided in the session below:

1) This system is only for demonstration purpose as a prototype for a blog system. No stylesheet has been applied. However, this gives the user more flexibility to design their own system. Secondary development is encouraged here as long as GPL is they obey the GPL terms.

2) There may be security issues when using this system. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY. System administers should regularly check the database to see if there are any potential risks, especially SQL injection attacks. You are also encouraged to re-develop the form submission programs to avoid such danger happening to you.

Enjoy using this blog system and do not hesitate to contact us when you find any problems or would like to offer any advice for improvement.

Niu Yunpeng
neilniuyunpeng@gmail.com

December 2016

My accomplishments

As I have said in the user manual above, this blog system is not for any commercial use. Therefore, I do not intend to make it look beautiful (and that's why I even did not use any .css file). However, I try to make myself learn as much as possible in this development circle.

I would like to highlight a few exciting points in this process below.

- Learn how to learn

Before I tend to try this project, I have learnt three programming languages: C, JavaScript and Pascal. All of them are acquired through formal courses and each of them takes about one semester (which is about 3 - 4 month).

However, in just past the ONLY 2 weeks, I have added all of the following programming languages into my skill set: HTML, CSS, PHP, MySQL and Markdown. So amazing, isn't it?

I have found a suitable method for self-learning: first going through a beginner tutorial; then checking out the documentation at the official website; after that, applying my knowledge into a project; finally, asking Google whenever I really cannot solve a problem on my own.

This pathway has been proven to be effective and efficient in learning programming. Meanwhile, this will be beneficial for my future study.

- Model-View-Controller (MVC) Framework

When I first started this project, I felt quite confused and did not know where to begin. MVC came into my view and I tried to organize my folder like that.

There are four sub-folders under my source code folder, namely, config, model, view and source. In the "config" component, all the configuration information is stored there, including the database server address, database name, username and password. They can be changed by the system administer to satisfy different environment. In the "model" component, it includes all the useful functions that deals with users' request and interacts with the database (although it is called "model", it may do the work of "controller"). In the "view" component, it controls the interface that is visible to users. In the "source" component, all the static pages for all articles are stored here.

In this way, the ease of maintenance and code reusability is improved a lot.

I have encountered a lot of problems and spent a lot of time solving them. I would like share two of them here.

- _SESSION super global variables may get lost magically.

This has been a known bug for PHP. However, a few small tricks may be useful:

1) Check that in your PHP installation configuration, session.use_trans_sid=1;

2) Sends the session_id as a _GET variable when redirecting to a new page happens.

- Make a confirm window before the delete operation is done

To do this, a JavaScript function `confirm()` will be utilized. However, it must be written as an `onclick` attribute instead of `onsubmit` attribute. Otherwise, the form will already be submitted and operated by the PHP script. Also, it has to be written in the form of "`return confirm(…);`". The `confirm` keyword is compulsory.

I really learnt and made progress a lot by developing this blog system. And more significantly, I understand how to think in a big picture when I need to develop such a complicated system instead of a single program.