Massive Content Management System

Anders Bergström       Björn Lindqvist       Roine Edmundsson
   c95abm@cs.umu.se       dva95blt@cs.umu.se       c99ren@cs.umu.se

13th June 2004

Abstract

“Massive content management system” is a web based system that gives the editor of a web page an easy to use and consistent interface when editing web pages. This article describes the concepts and ideas behind this system and also implementation details.

The system has not yet been tested in a production environment but tests has shown its functionality.

1 Introduction

Large web sites can be hard to keep up to maintain, not everyone knows how to write html or some other language that the pages are written in. And even if the editor of a certain page have that kind of knowledge he might not have the permissions to edit files on the server where the data for the home page is located. A content management system is a tool that makes the administration and editing of home pages easier by giving the editors (the people responsible for certain parts of a web site) a framework with predefined page layouts and a easy to use user interface when editing the pages. Large web sites often have different people responsible for different pages and because of that users should be able join different groups, giving them rights to edit certain pages and add new pages under ones they have rights to change. There should also be possible to give certain users rights that override the group rights, resulting in a safer system where users can be given limits to what they can do or more rights to do things if they have proven to be reliable.
2 Approach

The solution is based on a database running PostgreSQL 7.4 in which all the data for each page and surrounding information (users, page types and such) is located. To access this data PHP5 is used and to show the result on a page PHP5 is used to generate XHTML. Cascading stylesheets (CSS) have been used to layout the page. When the foundation was chosen the database was designed and a general flow of how the system should work was created.

The tools that are used have all been chosen due to they are free to use, well documented and widely used in the industry. This is perhaps not the case with PostgreSQL but it was chosen since it was what the group members had experience with. The choice of XHTML and CSS to present the user interface was the easiest since it is an open standard that is widely supported and used and will work with any standard compliant browser independent of platform.

The PHP session variable was decided to be used to verify that a user was logged on to the system, this to produce a safe environment where the password management was handled since the session variable is handled by the web server.

The most complex part was the design of the database, there were at least as many ideas as people involved in the project. This also lead to redesigning several times during implementation. What we wanted was a modular design that easily can be expanded to include more features. We also want the system to be easy to use, not only to the users but also for the implementators of page templates (Page types). Hence, it should not be necessary to know the details of the system to implement new page templates. To get some kind of structure of the website a parent/child relation exists between pages where the root page have no parent.

The permissions are set on a per page basis. At creation, a page inherits it’s parent’s permissions. Users are given different permissions for a page depending on which user group(s) they are a member of. There are two
system defined user groups, admin and editor. All users are members of the editor group. A member of the admin group always have permission to do everything. It’s also possible to set personal permissions for a specific user on a page. The personal permissions should override the group permissions for the user.

To handle large binary objects in the database, PostgreSQLs bytea data-type was chosen. It seems superior compared to use OIDs with import and export functions which breaks constraints in the database. This happens since the actual data is not held in the same table but in a special table for binary data.

3 Results

The resulting system is built by over 4000 rows of mixed PHP and XHTML code and the database is made up of 11 tables. It has been tested with small testpages. A screenshot of an example page can be seen in figure 1.

![Screenshot of santa’s page.](image)

Figure 1: Screenshot of santa’s page.

The main adminpage can be seen in figure 2.
The performance has not been measured but should not be a problem due to that we do not use any complex queries in the database or complex algorithms in the code.

4 Discussion

The biggest problem have been to communicate within the group, things have been misunderstood and because of that things have had to be re-designed and done one more time. Another problem have been that the real work started quite late and that made the product shown at the seminar not fully feature complete. The database design have changed alot during the project and one thing that is learned is that good design in the beginning saves time in the end.

The tools we have used has caused us little problem, but PostgreSQLs poor documentation is worth mentioning since it makes it hard to use. Especially if advanced concepts such as stored procedures or rare datatypes like bytea which is almost undocumented are to be used.

Another thing worth mentioning is the tight coupling of the PHP code,
XHMTL and the database which makes it hard to make a good and extensible design. There should be another layer that separates the user interface from the application logic.

5 Conclusion

We have succeeded in creating a content management system with the features that we set as a goal in the beginning. The final result is a web based system that allows users (editors) to edit web pages and administrators to give different users different rights.

A conclusion from the project is that the database access and the user interface should really be separated from each other to allow changes in the different parts without the need to rewrite the whole application.