GLUEX WEBLOGGER NOTE

Sean Sunley

SPARRO Group

Dept of Physics, University of Regina

Regina, SK, S4S 0A2

ABSTRACT

To develop/adapt/implement a web-logging (blogging) system for the GlueX Portal website. Requirements for the system were that it be absolutely free (subject to the usual terms of the GNU public licensing agreement), have a clean and simple interface, support multiple authors and categories, and support remote posting through the use of "blog client" applications. It was also deemed preferable that this system be as seamless as possible with the existing Portal infrastructure and be written in PHP.

INTRODUCTION

WEB-LOGGING

Web-Logging, or the more common term "blogging" is the process of using a web-based system to maintain an on-line journal with varying degrees of access. An entire virtual industry of blogging applications has sprung up around this simple idea. There are complete installable packages (often called Content Management Systems (CMS) however this term has a somewhat broader meaning as well) that turn a website into a blogging utility, as well as a whole host of different "blog client" applications which allow authors of blog sites to post to their blog without actually logging on to the website itself. Additionally, many of these CMSs have taken advantage of other existing technologies such as the XML based RSS. Readers can then point their news-reader at the RSS feed of their favourite blog and receive a continuous update of new posts, again without having to visit the site itself. This allows readers to visit only when they need to and to easily select only the articles/blog postings they are interested in.

The existing GlueX portal website which is powered by phpNuke (a more full-spectrum CMS) contains a very simple blogging system. It lacks many features that more popular blog systems have today such as multiple categories, varying levels of access, posting from blog clients and generation of RSS feeds. It was decided that this blogging system was inadequate for the needs of the collaboration and a more full-featured electronic logbook was required.

INITIAL ASSESSMENT:

After a quick initial assessment, the following steps were determined necessary to achieve the goal:

- **Education:** To learn as much as possible about existing blogging systems and software so as to be able to make more educated decisions.
- **Preliminary Search:** Investigate as wide a variety as possible of existing packages to determine which, if any, would suit our needs most appropriately with the least amount of work required, based on reviews and existing implementations and documentation.
- **Detailed Investigation:** If such a package was found, to further investigate its potential by installing it on the existing server and identifying exactly what would be required to adapt it to our needs.
- **Last Resort:** If no package was found, to identify the feasibility of creating such a package from scratch.
- **Implementation:** Once the path was determined and all requirements were known, actually implement the blogging system.
- **Documentation:** After implementing the system and verifying its operation, a FAQ for the portal must be written on usage with blog clients.

REPORT

EDUCATION:

From coming at this with very little kno wledge of blogging, I had quite a base I needed to build up before I could adequately understand the differences between the various packages available. The initial education phase lasted approximately 1 week however I continued to gain knowledge through later steps as well.

PRELIMINARY SEARCH:

The preliminary search took longer than I would have liked. It lasted close to another week as there are a huge number of blogging packages on the market and many have a severe lack of documentation. I was still learning more and more but my lack of experience in the area made things take longer than they may have otherwise. Impeding my progress (both in this and the previous step) was that there seemed to be a lot of confusion as to what certain terminology meant. Even the term "blog" was being interpreted and used with various conflicting definitions.

After a few days, I had decided on two packages that seemingly met our criteria. CMS Nucleus, and Drupal. As Drupal was a somewhat larger system than we actually wanted to employ, I decided to proceed to the next step using Nucleus.

However, after only a day of deeper investigation with Nucleus, I determined that it very much was not what the package advertised itself to be. Not only was it rather unwieldy to modify, but it did not support everything it said it did.

I was very reluctant to move on to Drupal however as it is really intended for full websites…much the same as the phpNuke system already employed by the Portal. I hit the web again and very quickly discovered WordPress…a system I had somehow managed miss previously. WordPress appeared to meet all criterion so I entered the second stage of testing with it.

DETAILED INVESTIGATI ON:

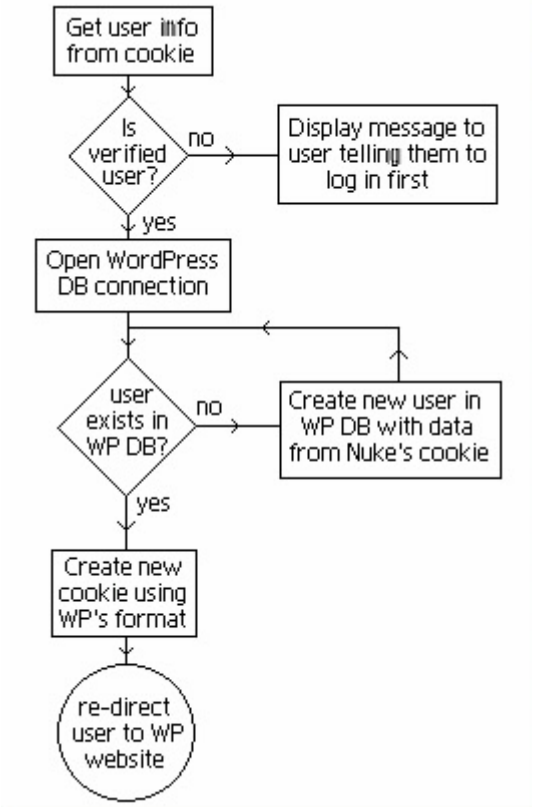
Installing and configuring WordPress to work standalone was very simple - it was operational within an hour. I tested its capabilities as far as blog operations went and all was well. Then came the more difficult task of identifying the exact method required to integrate WordPress with the existing phpNuke portal.

Nuke's modular interface worked to advantage. It was possible to create a new module that would take care of the interfacing back and forth between WordPress and the Portal through the use of cookies. Both Nuke and WordPress use cookies to store information about users who are logged in. This would allow the seamless integration of WordPress into the Portal.

IMPLEMENTATION:

The first task was to build this integration module on the development version of the Portal. From scratch, I wrote a Nuke module (in PHP) that created an instance of the WordPress database object. The module (now called **Logbook**) would then follow the process outlined in the simple flow diagram. The code for this flow chart is contained in **Appendix B**.

You can see how the module will create the user based on the exact data Nuke has stored for them. New users never have to register as they would ordinarily and the entire process of logging in and out of WordPress is entirely seamless. Neither is the security of the data in either database compromised as information is not actually transmitted except to store a cookie for already verified users.



After completion of this module code, some minor tweaks were needed to the WordPress interface to further make it feel like a part of the GlueX Portal. As well, some additional functionality was turned on and the presentation of the site cleaned up and made more attractive and consistent.

The migration to the primary (non-development) version of the Portal was relatively painless however some issues were discovered regarding the use of blog clients through the domain "portal.gluex.org". At the time of the writing of this document, these issues have not been resolved however the use of these clients is still fully operational by pointing the clients directly to the server itself (tantalus.phys.uregina.ca).

DOCUMENTATION:

The GlueX Portal's FAQ section contains several entries on usage with blog -clients. One blog client was recommended for each major platform (Mac, Windows, and Linux) and ins tructions on how to configure it for use with the Logbook was included. **Appendix A** also contains general usage information.

CONCLUDING THOUGHTS:

The entire process, while somewhat slow to get moving, went quite smoothly overall. The portal now has a fully operational blog system that meets all objectives. Meanwhile, I have expanded my knowledge of CSS, PHP, SQL, and the entire blogging industry quite significantly. Much of this knowledge I have been able to immediately apply to other personal projects as well as to professional projects such as the re-design of the University of Regina Department of Physics website (my current assignment).

APPENDIX A: GENERAL USAGE

ACCESSING THE LOGBOOK

Blog clients aside, the simplest method to access the Logbook is to first log in to the GlueX Portal website, look under the sidebar heading **Modules** for an entry called **Logbook**. Click on this link. Provided you have logged into the portal correctly, you will be taken to a new screen containing the Logbook's system.

READING POSTS

The first screen you see when logging into the Logbook contains summarized versions of the most recent posts in reverse chronological order. You can click on the title of any such post to read the full entry and add your own comments.

There are other ways to access posts as well. The menu bar on the right side of the screen contains numerous sections. You can organize and display posts by category, by author, and by date (using the calendar). And of course you can use the aforementioned newsreader as well.

AUTHORING POSTS

Anyone with a GlueX Portal account may use this electronic Logbook. To write a post of your own, click **add a post** under the menu heading **Other**. This will take you to a new screen where you can write your post. To actually post what you write, click **Publish**. You can also **Save as Draft** for future publishing if you are unable to complete the post in one sitting. When you are done, you can use the links at the top of the screen to navigate back to the Portal (Logout) or back to the main Logbook screen (Back to Journals).

LOGIN / LOGOUT

Ordinarily, you should not ever need to worry about logging in or out. The process is entirely automated and has been made seamless with the Portal. If you are properly logged in to the GlueX Portal, you will be logged in automatically. When you click the **Back to Portal** link, the logout process is taken care of for you.

APPENDIX B: LOGBOOK MODULE CODE

[The following code is the portion of the Logbook module which was entirely authored by myself. It is not a complete copy of the file which is mostly made up of a shortened version of the WordPress database module code. Both WordPress itself and the database code in the Logbook module were both modified with numerous tweaks and adjustments.]

```
<?php
//Following code was authored entirely by Sean Sunley
////////////////////////////////////////////////////////////////
//  create_wp_user
/*************************************
this function loads the user information from the NukeDB and create
the lines necessary in the WP DB returning TRUE if it succeeds or
FALSE on any failure.
*************************************/
function create_wp_user($nuke_userid, $nuke_login, $nuke_pwd, $wpdb) {
	//We have the user ID for this user from nuke but we want all their user information:
	global $db;
	require_once("mainfile.php"); //needed to access Nuke's database information and such
	$sql = "SELECT name, username, user_email, user_password FROM nuke_users WHERE
username='$nuke_login'";

	$result = $db->sql_query($sql);

	$row = $db->sql_fetchrow($result);
	if ($row == false) { return false; }

	// $row NOW CONTAINS OUR USER
	$user_ip = $_SERVER['REMOTE_ADDR'] ;
	$user_browser = $wpdb->escape($_SERVER['HTTP_USER_AGENT']);
	$user_email = $row[user_email];
	$now = gmdate('Y-m-d H:i:s');

	$result = $wpdb->query("INSERT INTO wp_users
		(user_login, user_pass, user_nickname, user_email, user_ip, user_browser, dateYMDhour, user_level,
user_idmode)
	VALUES
		('$nuke_login', '$nuke_pwd', '$nuke_login', '$user_ema       il', '$user_ip', '$user_browser', '$now', '1',
'nickname')");

	if ($result == false) {
		return false;
	} else {
		return true;
	}

}
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
//  verify_wp_user
/*************************************
verify_wp_user is a function that will identify whether a user exists in the WP DB.
If they do, it will log them in by creating a cookie and return "1".
If they don't, and $create='create' then it will attempt to create the user and make a cookie to log them in
	subsequently returning a "1" on success or 2 on failure.
If anything else fails, it will return a "3".
*************************************/
function verify_wp_user($nuke_uid, $user_name, $user_pwd, $create) {
	$wpdb = new wpdb(DB_USER, DB_PASSWORD, DB_NAME, DB_HOST);
	$query = "SELECT ID, user_login, user_pass FROM wp_users WHERE user_login = '$user_name'";
	$login = $wpdb->get_row($query);
	if (!$login) {
		if ($create == 'create') { //check to see if we should be creating on failed login attempt
```

```
                        if (!create_wp_user($nuke_uid, $user_name, $user_pwd, $wpdb)) {
                                return 2; //failed to create new user
                        } else {
                                //Successful creation of user
                                $login = $wpdb->get_row($query); //try to log them in again
                                if (!$login) { return 3; } //dies if second login failed
                        }
                } else {
                        return 3;
                }
        }
        //If we haven't failed by now, $login must create the
        //necessary information to construct the cookie for WP.

        //Construct cookie
        $cookiehash = md5("http://tantalus.phys.uregina.ca/gluex/wordpress");
        //echo "wordpressuser_$cookiehash, $login->user_login, " . (time() + 31536000) . ", " . COOKIEPATH;
        //echo "<br>wordpresspass_$cookiehash, " . md5($login    ->user_pass) . ", " .      (time() + 31536000) . ", " .
COOKIEPATH;
        setcookie('wordpressuser_' . $cookiehash, $login->user_login, time() + 31536000, COOKIEPATH);
        setcookie('wordpresspass_' . $cookiehash, md5($login->user_pass), time() + 31536000, COOKIEPATH);
        return 1;
    }
    ////////////////////////////////////////////////////////////////////



    ////////////////////////////////////////////////////////////////////
    //Loads the header frame for Nuke when an error occurs and we have to display it.
    function NukeFrame() {
       require_once("mainfile.php");
       $module_name = basename(dirname(__FILE__));
       get_lang($module_name);
       include("header.php"); //Loads the Nuke framework
       $pagetitle = "- My Journal";
       echo "<center><font class=\"option\">My Journal</font></center><br><br>";
    }
    ////////////////////////////////////////////////////////////////////


    ////////////////////////////////////////////////////////////////////
    //  MakeForm

    /*************************************
    MakeForm will either create the link that will log a Nuke user into WP or it will can out when the user
    is not logged in to Nuke.
    *************************************/
    function MakeForm($name, $address) {
       global $user, $EditedMessage, $cookie, $anonymous, $notify, $notify_email, $notify_subject, $notify_message,
$notify_from, $prefix, $db;

       if (is_user($user)) {
          cookiedecode($user);
          $uid = $cookie[0];
          $name = $cookie[1];
          $passwd = $cookie[2];

    //    We have a verified Nuke user wanting to access WP.  So we need to check and see if this Nuke user has a WP
account.
          $result = verify_wp_user($uid, $name, $passwd, 'create');
          switch ($result) {
                case 1:
                        header('Expires: Wed, 11 Jan 1984 05:00:00 GMT');
                        header('Last-Modified: ' . gmdate('D, d M Y H:i:s') . ' GMT');
                        header('Cache-Control: no-cache, must-revalidate');
                        header('Pragma: no-cache');
                        header('Location: http://tantalus.phys.uregina.ca/gluex/wordpress/');
                        exit;
```

```
            break;
            case 2:
                NukeFrame();
                echo "<p><b>Unable to create new WP user.</b></p>";
                include("footer.php");
            break;
            case 3:
                NukeFrame();
                echo "<p><b>Failed to create login cookie for Wordpress</b></p>";
                include("footer.php");
            break;
        } //end switch $result
    } else {
//      This user is not a verified nuke user so we can't let them touch the WP stuff
        NukeFrame();
        echo "<p>You must login before you have access to the journals.</p>";
        include("footer.php");
    }
}

////////////////////////////////////////////////////////////////////////////

//  Thus begins the actual code:

    MakeForm($name, $address);

////////////////////////////////////////////////////////////////////////////


?>
```