UserPagefortuity

Introduction

I live in Marin County, an hour north of San Francisco, in a rural area with redwood trees and hiking trails. I develop on a Mac OS X machine (I'm nostalgic for NeXT) and run a Sun Cobalt RAQ550 server (Linux) out of a co-lo facility deep in the heart of Silicon Valley. (I'm told it's there; I've never seen it; as long as i can ping it I assume it exists).

In 1991, as a member of the original www-talk mailing list, i was an early participant in the development of the Web. Since then I've worked as a Java developer on web applications for large companies. I like PHP for my personal projects.

I got interested in tikiwiki when i saw it was among the top 25 projects on sourceforge (December 2003). I had heard of it before but was under the (false) impression that it was only a wiki. To me it's not a Content Management System, either (though obviously it can be used as one), because the CMS products I've encountered are primarily useful in insulating users from HTML or PHP coding, and tikiwiki is most useful to me because i can tinker with the underlying code. Based on my experience with Java web applications, I'd call it a web application framework. A very well-developed one with lots of pre-built high-level functionality (unlike Java web application frameworks).

I'm pleased with tikiwiki, especially the large base of support!

Development Notes

Project

12 December 2003

I'm evaluating tikiwiki after having worked briefly with the xaraya CMS. Features in xaraya are called "modules": I wrote a "dailyquotes" module for xaraya. I'm working on a "dailyquotes" feature for tikiwiki, to explore the tkiwiki development idiom.

Development Guide

The tikiwiki Documentation and User Manual has a section on development, titled "Adding new features to Tiki" (page 333 of the Tiki 1.6 manual at http://de.tiki.org/tikidoc/). It's very basic and doesn't go into any depth. The page HowToDev suggests ways to get help. Coming from the world of Java, I expected something like Javadocs to map the API and I found
http://de.tiki.org/xref-head/
http://de.tiki.org/dox-head/html/
(there's not much description but all the classes and functions are there).

Creating Project Files

Relative to xaraya, tikiwiki is not very modular: Files for "features" are scattered through several directories. Xaraya organizes all files for a "module" in a single directory and its subdirectories (templates and scripts implementing user and admin APIs). Tikiwiki organizes all templates in one directory; lib scripts are in another; sql scripts that set up the database tables are in a db directory. With xaraya, it appears to be easy to develop functionality specific to one's own web site, swapping in a single "module" directory after upgrading to a newer

xaraya release. Tikiwiki is more monolithic: customized functionality requires changing multiple files. TikiPackager proposes a solution but this is not implemented as of December 2003.

To begin development of a "dailyquotes" feature, I created the following files:

tiki/lib/dailyquote/dailyquotelib.php	a dailyquote class encapsulating the user and admin API
tiki/tiki-admin_dailyquote.php	script for Admin page
tiki/templates/tiki-admin_dailyquote.tpl	template for Admin page
tiki/tiki-dailyquote.php	script to display a daily quote for users
tiki/templates/tiki-dailyquote.tpl	template to display a daily quote for users

Setting Up Permission Control

To set up control of permissions to access the Daily Quote Admin page it appears I have to edit the file

tiki/db/tiki.sql	database schema

to add an "INSERT INTO...tiki_p_edit_dailyquote" permission element. For now, to avoid modifying the distribution tiki.sql file, I'll just "borrow" an existing permission element (for now, tiki_p_edit_cookies) and allow access to Admin Daily Quote if the user is allowed to edit "cookies."

Creating Database Tables

I looked at alternatives to modifying

tiki/db/tiki.sql	database schema

to set up my database table (so I don't have to modify the distribution tiki.sql file). Xaraya has an API for automatically creating needed database tables at the time a "module" is installed (new features are installed programmatically from the web admin GUI). There is no equivalent in tikiwiki: One just copies in new scripts and modifies the code in templates. To create the database tables I need, I added code to the tiki-admin_dailyquote.php script that calls an "init_database()" method in the dailyquotelib.php script. This code will be called every time the user calls the Admin Daily Quotes page (which is extra overhead) but eases development because I can change the schema during development without using phpMyAdmin or the mysql command line to alter my table. I use the ADODB data dictionary functions for this, specifically the ChangeTableSQL() function see http://phplens.com/lens/adodb/docs-datadict.htm. This method checks to see if table exists and if the table does not exist, adds the table. If the table exists, it generates appropriate ALTER TABLE MODIFY COLUMN commands if a field already exists or ALTER TABLE ADD $column if a field does not exist. I wonder why these ADODB functions are not used in tikiwiki?

Adding A Link To The Admin Page

With tikiwiki, each page can be viewed by entering the file name of the script that generates the page directly as a URL (for example, http://localhost/tiki/tiki-admin_dailyquote.php). That's not unusual (since most people who have created HTML pages would expect this) but some other frameworks or CMS systems are parameter-driven and there is no one-to-one correspondence between the filename of the script that generates the page and the URL that appears in the web browser. In tikiwiki, it's very convenient that one can find the script that creates a page by looking at the browser URL and conversely, one can enter the filename as part of the URL if

one can't find a link to the page.

I didn't want to type http://localhost/tiki/tiki-admin_dailyquote.php every time I tested the Admin Daily Quotes page, so I looked for a way to add a link to the tiki application menu. In the version of tikiwiki that I am using today (v1.8 from CVS), there are six different template files to modify to add the link, corresponding to different themes. I am using the the "moreneat.css' theme, so I modified the file

tiki/templates/styles/moreneat/modules/mod-application_menu.tpl	application menu module for the "moreneat" theme

It seems to me this is not optimal architecture. Shouldn't the menu items be in one place only, rather than duplicated among each theme? If a developer does not modify all theme files, a user could change a theme and lose menu items.

Exception Handling and Reporting

Oh boy. This is where PHP4 makes me miss Java.

In tikiwiki code, it is trivial to catch an error and show the user an error message if the error is in the script that generates a page (for example, http://localhost/tiki/tiki-admin_dailyquote.php). For example, this could be catching an invalid value in a form. The tikiwiki idiom looks like:

```
if ($someError) {
$smarty->assign('msg', 'This is a helpful error message');
$smarty->display("error.tpl");
die;
}
```

This redirects the user to view an error page, showing the error message, with full formatting and links.

I don't know a good way to display an error message generated in scripts that don't contain the $smarty object, for example, in my script dailyquotelib.php. The only obvious approach is to use the PHP function "die('some message')" which produces an ugly page with only the message and nothing else (no page formatting, links, etc.). Xaraya has its own exception handling which provides functions that can be called anywhere to display an error page. I haven't found anything like this in tikiwiki.

Obtaining the Logged-In User

It's often important to know who is the current logged-in user. For example, when updating a database record, you might want to note the ID of the user who made the change. This might be implemented in a function in a library script (or class method). You can obtain the current user login (for example "fortuity") and userID (for example, "2") like this:

```
global $user;
global $userlib;
$userid = $userlib->get_user_id($user);
die('user is '.$user.' with id '.$userid.');
```

Xaraya's code would look like this:

$userid = xarUserGetVar('uid');

Personally, I prefer the Xaraya approach, obtaining the value through a single API call.