Technical Explorations

by Keith R. Bennett

WordPress Administration with Ruby

Sep 23, 2012 • keithrbennett

(This article is about the *wordpress_config_parser* gem, whose project page is at https://github.com/keithrbennett/wordpress_config_parser.)

The Problem

I've just consolidated blogs, email accounts, and web site data from multiple hosting companies onto a single hosting account. The WordPress blogs are the most important assets, and I want a good backup plan for them.

After some research, I find that WordPress data consists of files in the file system (e.g. photos), plus data in a data base, usually MySQL.

For the files, I make the whole shell account a big git repository, and use a Git host on the cloud to be the *origin* repo.

For the database, though, it's not so simple. Most of the information online points to the use of the PhpMyAdmin web app to perform a backup. However, I want this backup to be automated, repeatable, and self documenting. I need something that can be run from the command line. What to do?

The Solution

My research indicates that there is a command line alternative, *mysqldump* (which is probably the command called by phpMyAdmin). mysqldump generates a text file containing all the SQL commands necessary to reconstruct the data base with the identical schema and data.

Mysqldump's parameters include user id, password, host, and data base name. These are already available in plain text in the blog's *wp-config.php* configuration file. I'd like to read that file dynamically, rather than copying the data somewhere else, or requiring that it be provided on the command line (which would make it visible in the shell's command history).

With this as a goal, I wrote the *wordpress_config_parser* gem, which reads the wp-config.php file and makes its values accessible in a trivially simple way:

```
>require 'wordpress_config_parser'
parser = WCParser.new('/Users/me/public_html/blog')
db_name = parser.db_name
# ...
```

This approach is then used to build the mysqldump command line dynamically.

For the output SQL file, since my whole hosting directory tree is one large git repo, I can use the same file name every time I call mysqldump, and git will store the changes in a way that are easy to inspect and reconstruct.

Since I'm scripting this task, after the SQL file is generated, I might as well include the git *add*, *commit*, and *push* commands as well.

It turns out that mysqladmin will include a date/time stamp in the SQL output file, so even if there are no changes in the data base, two successive runs will produce nonidentical files. At first, this seemed like an annoyance, but on further thought I realized that wouldn't be such a bad thing – it would demonstrate that at that later time, the data was *still* the same as at the earlier time, something that could be guessed at without the extra commit but never proven.

The script I'm currently using is in the project's sample_scripts directory here. It's a bit bare, but illustrates how to get things done. If you want to run it, be sure to read the *Assumptions* section in the comments, and change your environment or the script as necessary. Don't forget to gem install the gem:

```
>gem install wordpress_config_parser
```

Next on the to do list is automating the periodic running of this script with cron or a Ruby equivalent.

- Keith

Published with GitHub Pages