GNOME SlackBuild (GSB) Website Documentation

Chip Cuccio

11/24/2007

This document explains how the GNOME SlackBuild (GSB) website was developed, how it works, how to maintain it, etc. This document also contains other general site-related information for developers of the GSB project, as well as web/sysadmins assigned web-related duties. This document is "living" - in other words, a work-in-progress.

Contents

1 Introduction

The GNOME SlackBuild (GSB) website was initially developed by Chip Cuccio in early 2005 as a "static" website. However, it didn't take long before we realized that we needed the site to be a bit more dynamic. At the same token, I really didn't want to deploy (or write) a CMS-type of app that used some DB back-end. I wanted tiny, fast, and simple. Really, the only key points of the website that are dynamic, are the news, version numbers, and changelogs, and the project TODOs.

So I custom-wrote a tiny "CMS" that uses flat files as its "DB". It's tiny, portable (and has been moved around from server-to-server a bunch!), and is simple for anyone to update content. Since 2005, it's the same code that's been in use (with a bunch of updates and revisions), and it still works well to this day,

2 Why this document?

This document was written to help developers or web/sysadmins manage, maintain, and continue the active development of the GSB website. Because the site was custom-written, a good document explaining how everything works is necessary.

3 How the site works

3.1 Basic technical info

The GSB site's main "CMS engine" is written in PHP. The engine performs parsing/'includes' against basic HTML templates for the content/copy in the main sections of the website. There are some support scripts and programs, not directly related to content rendering, written in shell and PHP as well.

The GSB site was written with strict adherence to standards, as well as, with excellent accessibility and usability guidelines. The site renders it's output in XHTML 1.1 (modular), with MathML plus SVG extensions. Styling is handled by strict CSS 2.0, with no "IE hacks". Visitors of the site whom use IE will see the site just fine, but will not be graced with the styling the CSS renders. The site works perfectly in modern standards-compliant graphical browsers, as well as older legacy browsers. Additionally, the XHTML output renders beautifully in text-based browsers ('lynx', etc.) and screen-readers. I designed the site intentionally, with forward-looking standards and structure, but at the same token, it degrades nicely in non-graphical and legacy environments. Again, IE will see the site in a degraded text-only mode, simply because IE sucks ass, and I refuse to write CSS hacks so that it looks pretty in a shitty browser.

It is important to note, that since the site is rendered in XHTML 1.1, the content-type on the server side, serves the pages as "application/xhtml+xml". This creates even more strict-ness when writing web code - since one tiny malformation or syntax error will render the entire page unreadable by many browsers. *All pages, all the time, should validate as 100% pure, valid XHTML 1.1* - always!

The chosen "LANG" (or "charset") attribute used to serve the site, is UTF-8. No other character set will do. UTF-8/Unicode is great stuff, and it's the future.

All images on the site are in PNG format. No GIFs, no JPEGs, no TIFFs. Just PNGs. Larger documents are in plain text, PDF, or just HTML. Documents which were written in LaTeX or other typesetting apps, then rendered to other formats, must include the original source code in SVN. Even the images I've made have the source PSD files committed to SVN. Another good example is this very document: although it was exported to PDF, etc., it was written in LyX/LaTeX - hence, the source files are in SVN. *Nothing in this site is closed-source.* Ever. It's all in SVN for public review and consumption.

3.1.1 Code review

It is highly recommended that anyone who wishes to help with the website, 'checkout' the web code from SVN and perform a thorough code review. If you have a web server accessible locally, you can even use the GSB site on that to debug, work on, test, etc.

3.2 Style guide

I'll try and keep this short and simple. But proper structure and semantics is of utmost importance when maintaining the GSB site. Here is the basic style guide I've used when writing/maintaining the GSB site:

- <h1> tags are reserved for the main site title and header. *They cannot be used anywhere else*, and no one should have to modify the pre-defined <h1> header.
- <h2> tags are reserved for page and (main) section headers. Every single page will have at least one <h2> tag. These can be modified and/or defined.
- <h3> through <h6> tags are used as sub-sections and nested sub-sections, and *must* have a "parent" tag/section above it. For example, you cannot use an <h3> tag for a page or main section header. Since <h2> tags are reserved for page and/or main section headers, <h3> must be structurally below <h2>, and so on. For the news section, news article titles are always enclosed in <h3> tags - and it's automatically done by the news rendering engine (more on that later).
- Ordered lists vs. unordered lists:
  Ordered lists are normally used where chronology is important, or when prefacing text specifies an exact number of forthcoming items. If the intended content constitutes a list, *use a list.* Of course, when appropriate, lists can be nested - even mixed nests (both ordered and unordered).
- Paragraphs:
  Any stand-alone body of text should be enclosed in <p> tags. Line-breaks (
) should be used conservatively. The use of tags to enclose stand-alone

blocks of text is strongly discouraged. If text within a paragraph or other block-level elements needs markup or styling, use tags with class attributes.

- Definition lists:
  Definition lists should be used when specifying and defining terms, or when specifying figurative questions and providing answers (such as our FAQ on the website).

- Code and pre-formatted text:
  All literal code must be enclosed in <code> and <pre> tags. Short snippets of code should be enclosed in <code> tags, while larger blocks of code should be enclosed in <pre> tags. *Do not* enclose <pre> blocks of text within any other block-level element.

- HTML Entities:
  Escape HTML entities! For example, "<" must be coded as < - otherwise the page will be malformed and will not load in many user agents.

- URIs:
  Links should never be verbs (e.g.: no "click here"). Literal URIs, whether linked or unlinked should be enclosed within brackets - e.g.: <http://gnomeslackbuild.org>.

- Quotes:
  Short quotes should always be wrapped in <q> tags. Never wrap quotes in quotation marks when using the <q> tag! For larger blocks of quotes, enclose them in <blockquote> tags. Additional block-level elements are permitted and within <blockquote> elements.

  - Citing:
    When using <q> and/or <blockquote> tags and quoting something/someone, if possible, utilize the 'cite' attribute within the tag.

- Tables should only be used to present *tabular data*. They are never to be used to handle/render layout. Ever!

3.3 Site layout

The layout of the site is quite simple:

- Homepage
- Sections
  - Subsections

And that's about it. I try to keep sections only two levels deep maximum, so not to lose folks. There's no search facility on the site, because it's so small, and because Google can do it anyway.

4 Helping with the site

I won't get into detail as to how the site's CMS engine works, because as a web hacker, it's your job to figure it out. Folks not familiar with PHP/HTML/CSS/etc. should not express interest in maintaining the site - 'cause you'll just be denied. We need experienced folks to help.

But I will cover a few basic, yet important aspect of the site, where maintaining content and a few other things is ridiculously easy.

4.1 News

On the landing/home page, there's a news section (and it's also in it's own main section - for archival purposes). All of those articles are handled and maintained in a flat text file within the <docroot> of the site: "news.txt". News is displayed on the site in reverse-chronological order (newest entry first). It's very simple to edit/add news; by editing that one small news.txt file.

The file is a specially-formatted text file, and is parsed by some PHP functions to display nice and cleanly on the site.

The format of the news.txt file is as follows:

```
###
date-time
title of news entry
body of news entry
```

The "###" text, is actually a separator, so that the PHP news function knows to parse that as one entire "article". The body section of the news entry accepts HTML, but that is the only place HTML is used. URIs are automatically made as links. To reference an internal URI, one must use the following format:

```
<a href="/foo/">See the foo section</a>
```

Hint: Internal URIs /never/ use "http://" - ever. Reference them to the base/root of the site, as shown in the above example.

Just open the <svnroot>/web/news.txt file for actual live examples used on the site.

4.1.1 News style guide

The news articles actually have their own style, and it's consistent. It is important to note, that news articles are automatically enclosed within HTML block-level elements. However, you still must enclose blocks of text/paragraphs in block-level elements (mainly, "<p>"). Lists, are OK - but header tags are not. The site news engine will handle everything else for you.

4.2 Screenshots

Screenshots are placed in `‘<webroot>/screenies‘`. Then, a simple shell script takes care of resizing, creating thumbnails, and adding eveything to the screenshots page. After screenshots have been dropped into the proper directory run:

```
cd ../support-scripts
./make-GSB-thumbs.sh
```

And that's it. Commit the changes to SVN, sync to staging, and make sure the screenshots page/section looks and works OK (i.e.: New thumbnails should now appear on the page. Thumbnails should be clickable links to full-size images. Full-size images should be in the ballpack of ~800x600 resolution.)

4.3 Version numbers

Version numbers are used all over the site for our project. But they only need to be maintained/specified in one tiny file:
<svnroot>/web/versions.txt

The legend for the version numbers is explained in the header of that file. To reference version numbers in the site templates ("<svnroot>/web/content/foo.html"):

- `<?php echo $gsb_bin_stable_ver; ?>`
  References the GSB actual stable binary release (tracks GNOME ver.)
- `<?php echo $gsb_source_ver; ?>`
  References the GSB source release. ...and so on (see <svnroot>web/common/versions_inc.php for all variables).

4.4 Getting SVN commit access to the site

We don't just let any Tom Dick and Harry have SVN commit access to the site. You must prove yourself first by submitting patches, etc. Once we've learned to trust you and your work, you'll likely be granted SVN commit access to maintain the site.

Developing/maintaining the GSB website is a three step process:

1. Develop and commit to <svnroot>/web
2. Sync the <svnroot>/web code to the staging site for testing, review, and approval.
3. Migrate staging to production

Under no circumstances is the staging environment to be used as the development environment. Development occurs on your own server/workstations. Staging is where we test and approve before pushing to production.

Even to migrate the SVN web code to the staging site, you must be granted access.

To migrate SVN web code to the staging site, we use a web-based facility (written by yours truly): <http://admin.gnomeslackbuild.org>. It's locked down, and web developers we can trust will be granted access to the facility.

At the moment, code migrations to the production site are handled by me once it's been approved. I'm very picky about the site functionality and the look-n-feel, it's just the way it works for now.

5 Open to suggestions

I should close saying that I'm always open to suggestions. I may dismiss most of them, though. All kidding aside, I embrace ideas, feedback, and web/code hackers to help make and keep this site as cool as the project it hosts. Thanks.