xhtmail

Project documentation

Patrice Levesque
ptaff.ca

xhtmail: Project documentation

by Patrice Levesque

Table of Contents

List of Tables

List of Examples

Chapter 1. Quick startup recipe

1.1. Prerequisites

- **PHP4 (http://www.php.net/).** May currently work now with PHP5, one day it will indeed
- **mailparse (http://pecl.php.net/package/mailparse).** PECL extension to decode mail
- **mbstring (http://php.net/mbstrin).** PHP extension to handle multibyte characters
- **iconv (http://www.php.net/iconv/).** PHP extension to convert charsets (though if the list to archive is using english characters only this can be optional)
- **Mail_Mime (http://pear.php.net/package/Mail_Mime).** PEAR extension (`Mail/mimeDecode.php` is needed from this package, PHP4 doesn't ship `iconv_mime_decode` yet)

The mailing list archives must be constituted of separate files containing 1 mail message each with full headers.

1.2. Installation

1. The `shared_www` directory should be moved to a web-accessible location (and renamed if needed). This directory contains all images, scripts and stylesheets spicing up the web pages. In the next step, this web-accessible directory will have to be specified as `XHTMAIL_IMAGES_BASE_URI`.
2. `xhtmail.php` should be edited - on top of the file there are different configuration settings to edit.
3. Ready!

1.3. Usage examples

Example 1-1. Minimal commandline invocation

```
php xhtmail.php -n mylist -o outdir myarchives/*
```

The above command writes all archives in the `outdir` directory by processing messages from the `myarchives` directory. In the generated pages, the archive will be referenced as `mylist`.

Example 1-2. Simple commandline invocation

```
php xhtmail.php -a samples/after.simple.html -b samples/before.simple.html -n
mylist -t "Archives of mylist" -o outdir myarchives/*
```

The above command writes all archives in the `outdir` directory by processing messages from the `myarchives` directory, using `samples/after.simple.html` and `samples/before.simple.html` as templates. In the generated pages, the archive will be referenced as `mylist` and described as `Archives of mylist`.

Complete command-line reference is available in the appendix (#cmdlineref) and is also shown when **php xhtmail.php `--help`** is entered at a terminal.

1.4. Customization

- Coders that know even just a bit of PHP should look at `samples/before.php` and `samples/after.php` to see what's possible. A more advanced template engine could have been used, but let's keep things that should be simple, eh, simple.
- Sample files (found in the `samples/` directory) are provided as examples for wrapping the output.
- Tweaking the stylesheets is strongly encouraged. When still young, the author caught a disease called "Bad Taste" and it does not seem to fade away after all these years.

Chapter 2. Details about the project

What, where, why, when, how?

2.1. Incentive

The author uses Sympa 5 (http://www.sympa.org/) as a mailing list manager. Though its pure mail engine works fine, its web archives (generated with **mhonarc 2.6.5 (http://www.mhonarc.org/)**) seem deficient. Other engines were studied (LISTSERV 14.4 (http://www.lsoft.com/products/listserv.asp), Hypermail 2.1.8 (http://www.hypermail.org/), dbmarc 20030519 (http://sourceforge.net/projects/dbmarc/)) but it feeled like the world needed a new mailing list archiver. Why? they all exhibit many annoyances from these:

1. **Cumbersome navigation.** "One message, one page". Ever searched google and found *the middle of the thread* in a mailing list archive? Feels like the middle of nowhere!

   xhtmail puts all messages of a thread in a single page.

2. **Weak table-based markup.** Tables are meant for tabular data; every web coder should know. Though in 1995 no other method existed to obtain a complex layout, this era is long gone.

   xhtmail generates semantic markup, with no tables; signatures, titles, dates, quotes in mail messages are recognized most of the times and wrapped as such in appropriate tags.

3. **No markup language metadata.** Most browsers and search engines crave the metadata stored in a web page.

   xhtmail recuperates most metadata from the mail message headers; for indexes and threads, navigational metadata is generated.

4. **Plain look.** Everybody knows how the web looked like in 1994. Monochrome monitors lost popularity. Most of the engines don't allow any control of the presentation.

   xhtmail exclusively uses CSS for presentation so the web archive can be easily styled; the engine adds icons and even X-Faces to the message, bringing life to otherwise plain messages.

5. **Whack-A-Mole URIs.** Archives can change at any time, messages can be deleted for different reasons. As most engines use an internal pointer to generate mail message URIs, should a page in the original archives be deleted, other URI will change.

xhtmail always generates the same URI for the same thread and the same anchor for the same message. Cool URI Don't Change™.

6. **Undescriptive URIs.** Each web resource should be described when possible; people (and search engines) like it though no RFC nor web standard ask for descriptive URI

   xhtmail uses a combination of timestamp and message thread subject to build the thread URIs, giving them expressiveness; as a bonus, webstats suddenly become more readable...

7. **Invalid markup.** Though W3C recommendations for HTML exist since 1995, only a fraction of the web markup shows validity and mailing list archivers too create tag soup.

   xhtmail complies with XHTML Strict recommendations and can be safely sent as `application/xhtml+xml`.

8. **Useless file extensions for messages.** As said before, Cool URI Don't Change™; if the mail archiver generates `.html` files, and suddenly the pages need to be wrapped in a scripting language, troubles arise.

   xhtmail puts all threads in separate directories so no visitor-visible extension is necessary.

9. **No/flaky support for feeds.** Many people like to stay informed of what's happening on a list but don't want to subscribe for different reasons.

   xhtmail exports RSS2.0 and Atom1.0 feeds with message summaries.

10. **Difficult integration with existing site templates.** It should be easy to get a uniform look through all pages of a website, even mail archives.

    *xhtmail is template-driven so existing bits of markup can be reused. Though the xhtmail code is written in PHP, any CGI language (**python**, **perl**, **ruby**, **php**, ...) can be used for templating.*

2.2. Methodology

Before anything, this script is really tied to the author's needs. Some limitations may seem stupid; contact the author, those probably arise from simple overlooks.

As a complement of the above remark: xhtmail was first built for a french-speaking mailing list; it will indeed work better with this than Thai or Arabic.

- As everything is sorted by timestamp before processing, no problem will arise when reindexing the same files over and over when there's new content. In fact, full reindexing is still needed to update the web mail archives. Filenames and relations will stay the same after each reindexing.
- When possible, URI should point to the English Wikipedia (http://en.wikipedia.org/). Two reasons: most URI won't change (which could be different for a software publisher's URI), and all pages will eventually be translated (and users of xhtmail can work as translators if they wish).

  Wikimedia (http://www.wikimedia.org/) was asked for a generic way to handle link translations (http://bugzilla.wikimedia.org/show_bug.cgi?id=2463) so for now links will just point to the english version. In the future a wrapper could be embedded in xhtmail if the previous bug is not resolved in a timely fashion.

- GNU **gettext** (http://www.gnu.org/software/gettext/gettext.html) was discarded for internationalization. Using **gettext** would mean scattering files all over the place; xhtmail should remain as self-contained as possible. **gettext** also requires locales for each of the translations to be installed, meaning trouble for many people that don't have root access. And as some differences with environment occur (LC_ALL? LANG? LANGUAGE?) using **gettext** would have meant spending too much time debugging setups. There are not so many strings anyway.

2.3. Performance

- Here on an Athlon 1600, about a thousand mail messages are processed in 30 seconds.
- Memory usage should remain minimal because only metadata about mail messages is kept through the processing; individual messages bodies and attachments are filtered directly file by file.

2.4. Caveats

- All output is UTF-8. After 40 years of trouble with non-english languages, no time to mess with weird charsets problems that UTF-8 usage solve. For pure english data, that makes absolutely no difference to US-ASCII, ISO-8859-X.

  At this moment, supported message input charsets include UTF-8, ASCII, ISO-8859-1, ISO-8859-15, win-1252, win-1256, win-1258.

  Contact author <`xhtmail.wayne@ptaff.ca`> or write a patch if you need other charsets - the author doesn't need them so no coding will happen without a request (especially when every problem is resolved when the mailing list users configure their mailer to use UTF-8).

- Mail messages *must* have a `text/plain` part or else they are simply skipped. `text/html` attachments are discarded as they are first netiquette-unfriendly, second hard to manage (xhtmail aims for pure XHTML on output and tagsoup parsing is not fun). Fortunately, some mailing list engines add a `text/plain` part to mail messages with only `text/html` prior to archival.

- Message URIs are built from the subject line and timestamp of the message. A clash will occur if two threads are started at the very same second with precisely the same subject line. xhtmail indeed is not designed for overactive user-support mailing lists where clueless lusers title all their mails "problem" and expect revelations.

2.5. Recognition

This software would probably not exist without the GPL and other collaborative licences. Bits and pieces were picked from different sources, I thank all those involved.

- Mail handling functions: PEAR (http://pear.php.net/) and PECL (http://pecl.php.net/) that make code reuse a snap
- Mimetype images: Horde project (http://www.horde.org/)
- OS images: AWStats (http://awstats.sourceforge.net/)
- MUA images: dispMUA (http://www.cweiske.de/misc_extensions.htm#dispMUA)
- Commandline parsing: bgetop (http://higginsforpresident.net/projects/source/getopt-0.1.phps)
- XHTML wordwrapping: Brian Huisman AKA GreyWyvern

Chapter 3. Helping out

Here are notes for the advanced usage; this chapter may be skipped by those who show no interest whatsoever about free software development.

3.1. API Documentation

Using **phpdocumentor** (http://phpdoc.org/), full phpdoc can be obtained for this code, by using a command:

```
phpdoc -dn xhtmail -f xhtmail.php -o HTML:frames:earthli -t api -ti "xhtmail
documentation" -s on
```

That doc is also available on http://ptaff.ca/xhtmail/api/.

3.2. Contributions

They're welcome. Simple patches can help the project get better in no time.

Any patch to xhtmail should be build with **diff** **`-ru`** **`distributed_xhtmail_directory`** **`patched_xhtmail_directory`** and mailed to <`xhtmail.wayne@ptaff.ca`> for revision. Only patches to a recent CVS version or the latest official release will be accepted.

3.2.1. Translations

Currently, only a handful of interface languages are supported. Adding a language should not take more than half an hour.

- A `t_LC` function needs to be added to `xhtmail.php`. The language code should be picked from ISO 639. `t_fr` is probably the most current language function so it can be used as a reference.
- XML entities *must* be used in the file to encode non-ASCII characters. That means numerical entities, and *not* HTML entities.

3.2.2. Detection of more MUA and OS

The set of detected MUA and OS is still small. Adding detection for them is simple when holding a mail message with their footprint.

MUA

- All the logic for MUA detection is found in the `extract_mua_from_email_header` function.
- After addition of a new MUA, a corresponding entry in the `get_mua_uri` function must be inserted.
- A 14x14 PNG icon should be added if necessary for the MUA, in the `shared_www/mua/` directory.

OS

- All the logic for OS detection is found in the `extract_os_from_email_header` function.
- After addition of a new OS, a corresponding entry in the `get_os_uri` function must be inserted.
- A 14x14 PNG icon should be added if necessary for the OS, in the `shared_www/os/` directory.

Appendix A. Complete command-line reference

Usage:

php xhtmail.php options file [file...]

Each file is a separate mail message, with full headers.

Options:

`-a FILENAME`
`--after=FILENAME`

What content (typically XHTML) should be put after the xhtmail output?

`-b FILENAME`
`--before=FILENAME`

What content (typically XHTML) should be put before the xhtmail output?

Special tags in the file will be replaced according to rules in Appendix B

No `<hN>` tags should be found in this file because they are already used by xhtmail for message titles.

`-c FILENAME`
`--contents-after=FILENAME`

What content (typically XHTML) should be put after the thread contents in thread pages?

`-e EXTENSION`
`--extension=EXTENSION`

What extension should we give to output files? Default is `html`. Web visitors won’t see this extension, this is intended for server-side processing (if your templates are written in **ruby**, use `rb`, for **python** use `py` and so on).

`-f NUMBER`
`--feed=NUMBER`

Number of Atom/RSS entries in the main archive output. Defaults to `0` (no feed)

See `-u` option

`-h`
`--help`

Displays this help text and exits

`-i FILENAME`
`--index_after=FILENAME`

What content (typically XHTML) should be put after the index in index pages?

`-l LANGUAGE`
`--lang=LANGUAGE`

What interface language should the generated pages use? Default is `en`

Available: en, fr

`-n STRING`
`--name_of_list=STRING`

Name of the mailing list for titling purposes

`-o DIRECTORY`
`--output_directory=DIRECTORY`

Where should the output files and directories be located?

`-p URI[,URI]`
`--picture_uri=URI[,URI]`

Image URI and icon URI respectively for the Atom/RSS feeds. Silently discarded if not used with `-f`; the icon should be the size of a favicon (16x16), if provided

`-t STRING`
`--title_of_list=STRING`

Long name of the archive (Like "`Archives of the foo mailing list`")

`-u URI,URI`
`--uri_base=URI,URI`

Base URI for webpages and RSS respectively, separated by a comma, like "`-u http://ex.com/path1,http://ex.com/path2`"; no slash is wanted at the end of the URIs. This parameter is silently discarded if not using the `-f` flag (feeds). These base URIs are used only for cross-linking between web pages and feeds, not for internal links. The feeds base should be *before* the feeds subdirectory (if your feeds are at http://example.com/feeds/example.xml, use http://example.com). If unspecified, the feeds will be located in the `feeds/` directory of the output directory (`-o`)

`-v`
`--version`

Output version information and exit

Danger Will Robinson

Option parsing is primitive at best. Should results be weird, command line inspection is strongly suggested.

Appendix B. Templates reference

Table B-1. Template tags

Tag	Meaning	Example substitution
<^ATOM_PATH^>	URI of the Atom feed	http://example.com/atom.xml
<^CONTRIBUTOR_NAMES^>	Comma-delimited list of people names who posted to the thread	Barney Rubble,Bamm Bamm,Betty McBricker
<^CREATOR_NAME^>	Name of the creator of a thread (who started the thread)	Fred Flintstone
<^DATE_OF_CREATION^>	Date of creation of the page (according to the mail messages timestamps)	2004-09-22
<^DATE_OF_UPDATE^>	Date of update of the page (according to the mail messages timestamps)	2004-09-23
<^FIRST_TITLE^>	Title of the first page in a set (title of the first month page on month pages and title of the first thread on thread pages)	mylist - 2008/01
<^FIRST_URI^>	URI of the first page in a set (first month URI in month index view and first thread URI in thread pages)	http://example.com/mylist/2005-09/
<^INDEX_TITLE^>	Title of the index page	Index
<^INDEX_URI^>	URI of the mailing list archive main index	http://example.com/mylist/
<^LAST_TITLE^>	Title of the last page in a set (title of the last month page on month pages and title of the last thread on thread pages)	mylist - 2008/12
<^LAST_URI^>	URI of the last page in a set (last month URI in month index view and last thread URI in thread pages)	http://example.com/mylist/2005-09/
<^NEXT_TITLE^>	Title of the next page in a set (title of the next month page on month pages and title of the next thread on thread pages)	mylist - 2008/07
<^NEXT_URI^>	URI of the next page in a set (next month URI in month index view and next thread URI in thread pages)	http://example.com/mylist/2005-09/
<^PARENT_TITLE^>	Title of the parent page	Parent

Tag	Meaning	Example substitution
<^PARENT_URI^>	URI of the parent page	http://example.com/mylist/2008-09/
<^PREV_TITLE^>	Title of the previous page in a set (title of the previous month page on month pages and title of the previous thread on thread pages)	mylist - 2008/05
<^PREV_URI^>	URI of the previous page in a set (previous month URI in month index view and previous thread URI in thread pages)	http://example.com/mylist/2005-09/
<^RSS_PATH^>	URI of the RSS feed	http://example.com/index.rss
<^TITLE^>	Title of the mailing list, given by the `-n` option	My list

Example B-1. PHP template

```
<?php
    // This next line will be changed by the templating system, so in fact
    // the $title variable will contain the substituted string.
    $title = "<^TITLE^>";

    // If the template variable is unavailable, the variable will still contain
    // the original template tag; here this is checked and if the title was not
    // replaced, change the title to 'Untitled'.
    if ($title == '<^' . 'TITLE' . '^>') {
        $title = 'Untitled';
    };
?>
```