Simple**PHP**Blog

Translation / Language Documentation and Guidelines

Updated: Saturday, September 09, 2006

Introduction
Thanks to many users around the world, SimplePHPBlog has become an international flavor. Submitted languages from countries west to east and north to south have expanded the user base substantially. One problem with getting these language files is maintenance. It can take hours to update the languages with each release. This documentation will help to alleviate that pressure, allowing our users base who can submit language updates to expand, allowing more time for the core developers to fix bugs and add new features.

Folder Structure
All translation files are kept in the root folder called languages. This folder must always be located in the root or the language files will not be found by the setup or install routines.

Each language or language-variant has its own folder. No files reside in the languages folder directly. The folder name should not include spaces, instead using the underscore character instead to keep with unix based file naming standards.

In each language folder, there are two files minimum: id.txt and strings.php. Id.txt contains the name of the language file as it will be shown to the user in the interface for the blog configuration. This file is a single line with no line feeds or carriage returns. The strings.php file is the location where all code in SimplePHPBlog goes to get its translation. That file is explained below. This is also the file that must be submitted when a language file is updated for inclusion in the releases.

How Can I Tell If My Language File matches the English one?
It would be a very tedious process if you had to figure out which translation items needed to be moved from the English file to your local translation. For this reason we have a utility available in every release of SimplePHPBlog. When you are logged into your blog, open the page called http://yourdomain/languages.php. Then pick the language files you want to compare and click Submit.

You will then be provided a listing of all files in the blog, showing any language items that may be missing (basically that one file has it but the other does not).

In the example above, the archives section is missing title, showall and menu_moderation. This is a good example of how the languages work: menu_moderation is a global variable in the file, so it won't be in the area marked by "case 'archives'." It is somewhere else in the file. However, title and showall should both by included, therefore archives might look like:

```
case 'archives': // New for 0.4.8
       $lang_string['title'] = "Archives";
       $lang_string['showall'] = "Show All";
break;
```

See the section below on Global, Limited and Specific strings to know where to look for each of these items; menu_moderation above is an example of a global string (it's not in archives) where title and showall are examples of specific strings.

Strings.php

The strings.php file is broken down into several sections that allow for individual areas of SimplePHPBlog to reference specific screens. The file is a regular PHP file, so it must ALWAYS start with <?PHP and end with ?>. Otherwise the file won't work.

> **Name, Description, Editors, Compatible Version**
>
> The first few lines of the translation are for explaining who last updated the file and the versions that are compatible with them. These are comments in the code, therefore they must all be prefixed with // like the following example:
>
> ```
> // English Language File
> // (c) 2004 Alexander Palmo, apalmo <at> bigevilbrain <dot> com
> ```

```
// Simple PHP Version: 0.4.9
// Language Version:   0.4.9.0
```

This introduction section can contain whatever text you wish as it has no bearing of how the translation file works.

Function name and global variables

Do not edit this section. It is required for the file to properly run.

Language Codes and ISO Character Sets

Your local server needs to know how this language is to be presented to the user. This section defines how it will look. If you do not understand what this means you will need to research more before creating a new language file – if you are editing an existing file, leave it as is.

$lang_string['language']: used internally in the blog software. This should match your language name.

$lang_string['locale']: This is an array of all of the locale codes that your server will use to properly display the language. You can put as many as required in the array. The most common use is that Unix and Windows have different locale entries (two letter vs three letter).

$lang_string['rss_locale']: The locale used for the rss feed only.

The html_charset and php_charset are defined so the server knows how to display the specific characters. From Wikipedia: "A **character encoding** or **character set** (sometimes referred to as code page) consists of a code that pairs a sequence of characters from a given set with something else, such as a sequence of natural numbers, octets or electrical pulses, in order to facilitate the storage of text in computers and the transmission of text through telecommunication networks. Common examples include Morse code, which encodes letters of the Latin alphabet as series of long and short depressions of a telegraph key; and ASCII, which encodes letters, numerals, and other symbols, both as integers and as 7-bit binary versions of those integers, generally extended with an extra zero-bit to facilitate storage in 8-bit bytes (octets)."

The line starting with setlocale must be included as is in order for the file to work properly.

Global vs. Limited Strings vs. Specific Strings

The strings.php file has three types of strings that allow for different usage in the blog:

Global Strings

These are at the top of the file after the charset/locale lines and can be used anywhere in the blog regardless of the page/screen. These are mostly used for button and menu translation.

Limited Strings
Limited strings are those that are not required globally, but are needed in more than one place. An example section might look like:

```
if ( $page == 'add' || $page == 'add_static' || $page == 'comments' ||
$page == 'add_block' ) {
                      $lang_string['label_subject'] = "Subject:";
                      $lang_string['label_insert'] = "Insert Special:";
                      $lang_string['btn_bold'] = " b ";
                      $lang_string['btn_italic'] = " i ";
}
```

Specific Strings
Strings only used in one screen/page are assigned within a case area in the php file. For example, if you are looking to add a translation line to the archives block, you would look for a section that starts with:

```
case 'archives':
```

and ends with:

```
break;
```

Everything that is in between is specific to the archives.php file.

Other Guidelines:
We try to follow guidelines when creating these files and they are pretty common to most development people, but not necessarily to those of you who wish to edit these files and have very little technical experience. Here is a short list of items to watch out for:

1) **Left alignment**: Follow the way the existing English file is laid out. Start of a block is always the most left of any text (for example: case). The $lang_string is indented from that, and the break; lines up with the start of the block.
2) **Long strings / paragraphs**: You can press enter on a long line of text before you end it to make the line visible without scrolling, etc. As long as the ending quotes and semicolon are at the end of the line, SimplePHPBlog won't care.
3) **Naming the language**: Make the name in the id.txt as generic as possible. Do not include the version in the name. That information goes in the block at the beginning of the strings.php file.
4) **Submitting Translations**: Submit all translations to Sourceforge in the Patches section. We get too many emails per day to recognize all the translations and bug fixes that come that way.

5) **Button Translation:** For text that is on a button, make sure that instead of using an actual space in the text, use   instead.

Again, we'd like to that everyone who submits translations, and we are most appreciative of the sentiment. We could do more translations ourselves, but the results you get from an automated translation engine like Google or Babelfish are questionable at best. You all make this a better piece of software ☺.

The SimplePHPBlog Development Team