Translations in PrestaShop 1.5

Table of contents

Translations in PrestaShop 1.5

File paths and descriptions

PrestaShop has several types of translations:

- Front-office translations
- Back-office translations
- Error messages translations
- Field name translations
- Installed module translations
- PDF file translations
- E-mail template translations

For each type of translation, PrestaShop's internal translation tool parses a specific set of folders in order to retrieve all the translatable strings it contains, and present them to the translator.

Front-office translations

The following folders are parsed:

- `/themes/name_of_the_theme/`
- `/themes/name_of_the_theme/override/` (new in 1.5)

The translatable strings use the following Smarty syntax:

```
{l s='There are no products in this category.'}
```

Two options can be added to this syntax:

- "`js=1`": indicates that this string is contained within JavaScript code.
- "`sprintf='any string or number'`": see the various examples in the "Sprintf" section of this document (new in 1.5).

All the front-end translations are stored in the following file, with `iso_code` being the ISO 3166-1 code (http://www.iso.org/iso/country_codes.htm) for the language of the translated strings (`de`, `fr`, `en`, `it`, `es`, etc.):

- `/themes/`**`name_of_the_theme`**`/lang/`**`iso_code`**`.php`

The translations are stored in a PHP array, named `$_LANG`, and take the following form, using an MD5 of the string:

- `$_LANG['address_19f823c6453c2b1ffd09cb715214813d'] = 'Champ requis';`

The identification key is built by combining the name of the template file from which the string comes, an underscore, and the MD5 hash of the string itself.

> ⚠ If the same string appears in several different controllers, it will appear as many times in the translation tool.
>
> On the other hand, if one string appears in both a controller and its template, it will only appear once for this controller.

Back-office translations

The following folders are parsed:

- `/admin/`
  - Specific files: `header.inc.php`, `footer.inc.php`, `index.php`, `login.php`, `password.php`, and `functions.php`
- `/admin/tabs/`
- `/classes/`
  - Specific file: `AdminTab.php`
- `/admin/themes/` (new in 1.5)
  - All the template files in all the sub-folders are parsed
- `/controllers/admin/` (new in 1.5)
- `/override/controllers/admin/` (new in 1.5)
- `/classes/helper/` (new in 1.5)
- `/classes/` (new in 1.5)
  - Specific file: `AdminController.php`

The translatable strings use the following syntaxes:

- Controllers and classes:

```
$this->l('Add new root category')
```

- PHP files at the root of the `/admin` folder:

```
translate('Customer name:')
```

- Back-office templates:

```
{l s='Add tag'}
```

Three options can be added to these syntaxes:

- "`js=1`": indicates that this string is contained within JavaScript code.

- "`slashes=1`": enables you to add backslashes (\) to a string.
- "`sprintf='any string or number'`": see the various examples in the "Sprintf" section of this document (new in 1.5).

All the back-end translations are stored in the following file, with `iso_code` being the ISO 3166-1 code (http://www.iso.org/iso/country_codes.htm) for the language of the translated strings (`de`, `fr`, `en`, `it`, `es`, etc.):

- `/translations/**iso_code**/admin.php`

The translations are stored in a PHP array, named `$_LANGADM`, and take the following form:

- `$_LANGADM['AdminAccess151648106e4bf98297882ea2ea1c4b0e'] = 'Mise à jour réussie';`

The identification key is built by combining the name of the controller from which the original string comes and the MD5 hash of the string itself.

Note that:

- All the templates from the `/admin/themes/default/templates/controllers/` folder are prefixed with "Admin" + the name of the file.
- All the files from the `/classes/helper/` folder are prefixed with "Helper".
- All the files at the root of the `/admin/themes/default/templates/` are prefixed with "AdminController".

> ⚠ If the same string appears in several different controllers, it will appear as many times in the translation tool.
>
> On the other hand, if one string appears in both a controller and its template, it will only appear once for this controller.

Error messages translations

The following folders are parsed:

- `/`: all the PHP files at the root of the site.
- `/classes/`
- `/controllers/`
- `/override/classes/`
- `/override/controllers/`
- `/admin/`
- `/admin/tabs/`
- `/modules/`: all the modules' files are parsed.
- `/controllers/admin/` (new in 1.5)
- `/controllers/front/` (new in 1.5)
- `/override/controllers/admin/` (new in 1.5)
- `/override/controllers/front/` (new in 1.5)
- `/override/classes/` (new in 1.5)

The translatable strings use the following syntax:

```
Tools::displayError('An error occurred while creating archive.')
```

By default, the string goes through the `htmlentities()` function, in order to convert special characters to HTML entities on the fly. You can specify that `htmlentities()` should not be used by adding a final option:

```
Tools::displayError('An error occurred while creating archive.', false)
```

> This method must not be used within modules, because then the string translations would be saved in the `/translations/iso_code/errors.php` file instead of the `/modules/name_of_the_module/translations/iso_code.php` file.

All the error messages translations are stored in the following file, with `iso_code` being the ISO 3166-1 code (http://www.iso.org/iso/country_codes.htm) for the language of the translated strings (`de`, `fr`, `en`, `it`, `es`, etc.):

- `/translations/iso_code/errors.php`

The translations are stored in a PHP array, named `$_ERRORS`, and take the following form:

- `$_ERRORS['00569f4db559dc94d9a954a090c22b85'] = 'Impossible d\'écrire';`

The identification key is built using the MD5 hash of the string itself.

> If one string appears in several different files, it will only appear once in the translation tool.

Field names translation

The following folder is parsed:

- `/classes/` and all of its sub-folders.

The string are those returned by the `getValidationRules()` method from `ObjectModule` for each class.

All the field names translations are stored in the following file, with `iso_code` being the ISO 3166-1 code (http://www.iso.org/iso/country_codes.htm) for the language of the translated strings (`de`, `fr`, `en`, `it`, `es`, etc.):

- `/translations/iso_code/fields.php`

The translations are stored in a PHP array, named `$_FIELDS`, and take the following form:

- `$_FIELDS['Address_2df2ca5cf808744c2977e4073f6b59c8'] = 'Téléphone mobile';`

The identification key is built by combining the name of the class from which the string comes, an underscore, and the MD5 hash of the string itself.

Installed module translations

> When in production mode, only the currently installed modules can be translated. If you wish translate all modules, whether installed or not, you must put your installation of PrestaShop in "debug" mode.

The following folder is parsed:

- `/modules/` and all the sub-folders for the various modules.

The translatable strings use the following syntax:

- For all PHP files:

```
$this->l('Add new root category')

...or...

$module->l('Add new root category')
```

- For template files:

```
{l s='Add tag'}
```

Three options can be added to these syntaxes:

- "`js=1`": indicates that this string is contained within JavaScript code.
- "`mod='blockcms'`": adds the name of the module.
- "`sprintf='any string or number'`": see the various examples in the "Sprintf" section of this document (new in 1.5).

> The `Tools::displayError()` method must not be used within modules, because then the strings' translations would be saved in the `/translations/iso_code/errors.php` file instead of the `modules/name_of_the_module/translations/iso_code.php` file.

Since PrestaShop 1.5, you can translate modules depending on the theme. Therefore, the list of translation files is as such:

- `/modules/`**`name_of_the_module`**`/translations/`**`iso_code`**`.php`: for the default theme.
- `/themes/`**`name_of_the_theme`**`/modules/`**`name_of_the_module`**`/translations/`**`iso_code`**`.php`: for any other theme.

The translations are stored in a PHP array, named `$_MODULE`, and take the following form:

- `$_MODULE['<{blockcms}prestashop>blockmobilecms_d1aa22a3126f04664e0fe3f598994014'] = 'Promotions';`

The identification key is built by combining the name of the module from which the original string comes (i.e. "blockcms"), following by the name of the theme (i.e. "prestashop"), followed by the name of the file (i.e. "blockmobilecms"), followed by an underscore, and finally the MD5 hash of the string itself.

> If one string appears in several different files, it will only appear in the translation tool.

PDF files translations

The following folders are parsed:

- `/classes/PDF.php`
- `/override/classes/PDF.php`
- `/classes/pdf/` (new in 1.5)
- `/override/classes/pdf/` (new in 1.5)
- `/pdf/` (new in 1.5)
- `/themes/name_of_the_theme/pdf/` (new in 1.5)

The translatable strings use the following syntax:

- For classes files, `ClassName::l()`:

```
HTMLTemplateInvoice::l('Invoice ')

...or...

HTMLTemplateDeliverySlip::l('Delivery')
```

- For templates:

```
{l s='Delivery Address'}
```

Two options can be added to these syntaxes:

- "`pdf=true`": adds the name of the module.
- "`sprintf='any string or number'`": see the various examples in the "Sprintf" section of this document (new in 1.5).

All the error messages translations are stored in the following file, with `iso_code` being the ISO 3166-1 code (http://www.iso.org/iso/country_codes.htm) for the language of the translated strings (`de`, `fr`, `en`, `it`, `es`, etc.):

- `/translations/`**`iso_code`**`/pdf.php`

> **In version 1.5**
>
> If you use the default theme, translations will be stored among the default translations: `translations/`**`iso_code`**`/pdf.php`
>
> If you use any other theme, translations will be stored in the theme's folder: `themes/`**`your_theme`**`/pdf/lang/`**`iso_code`**`.php`

The translations are stored in a PHP array, named `$_LANGPDF`, and take the following form:

- `$_LANGPDF['PDF_invoicecb5efbba6a6babef9082ca6976928ca7'] = 'Bon de livraison n°';`

The identification key is built by combining "PDF_invoice" (v1.4) or "PDF" (v1.5) with the MD5 hash of the string itself.

> If the same string appears in several different controllers, it will appear as many times in the translation tool.
>
> On the other hand, if one string appears in both a controller and its template, it will only appear once for this controller.

E-mail template translations

There are two types of translation when touching upon e-mail templates: the template itself, and the mail object/title. They are completely different things.

Mail objects

The following folders are parsed:

- `/classes/`

- `/controllers/`
- `/admin/`
- `/admin/tabs/`
- `/modules/name_of_the_module/`: if a module contains a `/mails/` folder, all of the module's files are parsed.
- `/controllers/admin/` (new in 1.5)
- `/controllers/front/` (new in 1.5)
- `/override/classes/` (new in 1.5)
- `/override/controllers/admin/` (new in 1.5)
- `/override/controllers/front/` (new in 1.5)

The translatable strings use the following syntax:

```
Mail::l('Your new admin password', (int)$id_lang)
```

The second parameter is mandatory. If it is absent, the mail will be sent with an object in the shop's default language.

All the mail translations are stored in the following files, with `iso_code` being the ISO 3166-1 code (http://www.iso.org/iso/country_codes.htm) for the language of the translated strings (`de`, `fr`, `en`, `it`, `es`, etc.):

- `/mails/`**`iso_code`**`/lang.php`: for the default translations.
- `/themes/`**`name_of_the_theme`**`/mails/`**`iso_code`**`/lang.php`: for each theme's translations.

The translations are stored in a PHP array, named `$_LANGMAIL`, and take the following form:

- `$_LANGMAIL['Welcome!'] = 'Bienvenue !';`

The identification key is built by using the original string directly.

Mail templates

The following folders are parsed:

- `/mails/iso_code/`: for the default e-mail templates.
- `/themes/name_of_the_theme/mails/iso_code/`: for each theme's e-mail templates.

The translated e-mail templates are those available in English. For instance:

- If the `/mails/en/account.html` file exists, then it will be translatable for each language enabled in the back-office.
- If the `/themes/name_of_the_theme/mails/en/account.txt` file does not exist, then it will not be available for translation in the back-office.

Translating the installer

By default, the installer in handful of languages: French, English, Spanish, Portuguese, etc. But you might want to help users from your country by translating the installer in a language that is not yet featured.

Let's work with Vietnamese, for instance.

⚠ The following is based on PrestaShop 1.5.4.0, some parts may not work for earlier versions of PrestaShop.

You must have PrestaShop extracted to some directory, for instance `/var/www/prestashop`. You should be able to access it the URL `http://localhost/prestashop`.

Put the necessary file structure in place

Make sure to follow this process:

- go to your install directory (`/var/www/prestashop/install`, or `/var/www/prestashop/install-dev` if you have a development version)
  - in the `/langs` folder, duplicate the `/en` folder and rename it to /vn
  - in the `/fixtures/apple/langs` folder, duplicate the `/en` folder and rename it to `/vn`
- in your `/langs/vn/img` folder, rename all the files substituting the `/en` prefix with `/vn`: for instance `en-default-category.jpg` becomes `vn-default-category.jpg`.
  If you use Linux, you could for instance run the following command:

```
for i in en*; mv $i `echo $i | sed 's/en/vn/'`
```

ⓘ The language code of Vietnamese in PrestaShop is "vn", not ISO 639-1's "vi". PrestaShop sometimes deviates from the standards for various reasons, but now it is too late to change everything.

Therefore, you should first check that you are indeed using the correct language code:

- The main languages are listed here: http://www.prestashop.com/en/translations. The correct code is in the "ISO CODE" column.
- If your language is not in the list, you can use any two-letter code from the ISO-639-1 list, as long as it is not already used.

In any case, you should first contact PrestaShop's translation team to make sure you do not work at the same time as another contributor.

Setting the details for the new language

Edit the file `/langs/vn/language.xml` and put the appropriate values. Instead of:

```
<?xml version="1.0" encoding="UTF-8"?>
<language>
  <name><![CDATA[English (English)]]></name>
  <language_code>en-us</language_code>
  <date_format_lite>m/j/Y</date_format_lite>
  <date_format_full>m/j/Y H:i:s</date_format_full>
  <is_rtl>false</is_rtl>
</language>
```

...you need to use the corresponding values in the language:

```
<?xml version="1.0" encoding="UTF-8"?>
<language>
  <name><![CDATA[ting Vit (Vietnamese)]]></name>
  <language_code>vn-VN</language_code>
  <date_format_lite>d.m.Y</date_format_lite>
  <date_format_full>d.m.Y H:i:s</date_format_full>
  <is_rtl>false</is_rtl>
</language>
```

Inside the `<name></name>` tag, the first word is the language name in that language itself, the second word (in parenthesis) is the language name in English.

At this point, the only missing thing is the country flag. Download the free FamFamFam Flag Icons pack, and find the flag for the language's main country (or most representative). Use the `.png` version of the file.

Here is what the installer looks like now:

At this point, you should perform a full installation and check that it completes successfully. If it does not, you probably did something wrong, such as forgetting to rename all the images... Please go back up and check everything!

Starting the translation of the installer

If the installation succeeded, you can start translating! Here is how:

- Go to `http://localhost/prestashop/install/dev/translate.php`, select "vn" in the dropdown, and translate! The "Submit" button is at the bottom of the page. Save your work regularly.
- Once this is done, open the file names `/langs/vn/install.php` with a text editor, and complete the few translations that are not present in the interface : search for "`menu_welcome`", "`menu_license`", "`menu_system`", "`menu_database`", "`menu_configure`" and "`menu_process`". They are usually at the beginning of the file. Take care of escaping all the single quotes with backslashes if you use some in the translation: "Terry /'s". Adjust the information block if needed.

Last steps

There are many things we did not translate yet, they are contained in the XML files:

- `/langs/vn/data/*.xml`: those are the most important, open them in a code editor and translate the contents (everything that looks like real English sentences, not the tags or attributes names!). Pay attention not to translate anything that is an ID! If unsure what to translate in those files, ask away!
- `/fixtures/apple/langs/vn/*.xml`: same principle as above, these are the translations of the demo products.
- There are images in `/langs/vn/img`. Some of them contain English text: edit them with your favorite image editor.

> Try installing PrestaShop regularly while you translate, so that if the installation fails you will find more easily what change caused it to fail!

Once you have done all this please share your work with the community by contacting translation@prestashop.com, or just make a pull request!

Sprintf

The `sprintf()` function, which is standard to PHP, must be used in the correct way throughout PrestaShop.

In PHP files

Correct way:

```
sprintf($this->l('Empty string found, please edit: "%s"'), $string);
sprintf(Tools::displayError('Please create a "%1$s.php" file in "%2$s"'), $string1, $string2)
```

Incorrect way:

```
$this->l('Empty string found, please edit:').'"'.$string.'"';
Tools::displayError('Please create a').' "'.$string1.'.php" '.Tools::displayError('file in').'"'.$string2.'"';
```

In template files (new in v1.5)

Correct way:

```
{l s='renamed the /admin folder (e.g. /admin123%d)' sprintf=$number}
{l s='renamed the /%1$s folder (e.g. /admin123%2$d)' sprintf=[$string, $number]}
```

Incorrect way:

```
{l s='renamed the /admin folder (e.g. /admin123'}{$number})
{l s='renamed the'} /{$string} {l s='folder (e.g. /admin123'}{$number})
```