System Administrator Guide

System Administrator Guide

This guide will help you configure a better and safer Web server.

Once this is done, you will be ready to install PrestaShop, using our Getting Started guide: http://doc.prestashop.com/display/PS16/Getting+Started .

PHP configuration

Manipulating php.ini

Many of the advices in this guide require you to edit the `php.ini` file, found in your server's PHP install folder (not in PrestaShop's folder).

Not all hosts will allow you to edit or even access this file, so contact your host if you cannot access it.

For instance, you probably won't have access to `php.ini` on a shared hosting. If your host doesn't provide the required configuration by default and you cannot touch `php.ini`, then you should either move to a dedicated hosting, or change to a more permissive host.

Still, editing `php.ini` remains a technical and advanced action. If your shop does currently work well, there's no need for you to touch that file, let alone change host.

Editing the PHP configuration requires you to change some values in the `php.ini` file, most of the time from "On" to "Off" or vice versa. The file contains a lot of documentation for each line: be sure to read the ones pertaining to your changes, in order to better understand them. Be careful of what you edit, as this has a direct impact on the way PHP runs, and therefore on your servers stability and even security.

Required settings

In order for PrestaShop 1.6.x to run properly, your PHP installation must feature the following settings and libraries:

- MySQL (or Percona Server, see "Improving PrestaShop's performances" section below).
- GD library.

- Dom extension.
- `allow_url_fopen`.

The MySQL extension enables to access your data. PrestaShop simply cannot work without it.
You can also use the drop-in replacement Percona Server, which offers better performance than the standard MySQL server.

The GD library enables PHP to dynamically manipulate images. PrestaShop uses it to resize and rework the image files that are uploaded (watermarking, trimming, etc.). Without images, an online shop loses most of its interest, so make sure that GD is enabled!

The Dom extension enables to parse XML documents. PrestaShop uses for various functionalities, like the Store Locator. It is also used by some modules, as well as the `pear_xml_parser` library.

The `allow_url_fopen` directive enables modules to access remote files, which is an essential part of the payment process, among others things. It is therefore imperative to have it set to **ON**.

In short, it is imperative to have the following directives set to the indicated values:

```
extension = php_mysql.dll
extension = php_gd2.dll
allow_url_fopen = On
```

Recommended settings

Your PHP installation should feature the following settings and libraries, for best experience:

- GZIP support.
- Mcrypt library.
- `register_globals` disabled.
- `magic_quotes` disabled.
- `allow_url_include` disabled.
- Safe mode disabled.

Having GZip support enables the web server to pack web pages, images and scripts before sending them to the browser. This makes navigating the shop faster, and therefore a more agreeable experience.

The Mcrypt provides PHP with a hardened security layer, which enables the use of more hashing and cryptography algorithms.

The `register_globals` directive, when enabled, defines all environment variables (GET, POST, COOKIE, SERVER...) as global variables. **It is unsafe to use unset variables**, because a user could easily set a value into this variable by using the GET method, for example. It is therefore imperative to set `register_globals` to **OFF**.

The `magic_quotes` directive automatically escapes (or "adds antislashes", see http://php.net/manual/en/function.addslashes.php) to all special character sequences (`'`, `"`, `\`, `NULL`) for all environment variables (GET, POST, COOKIE, SERVER...). This option must be set to **OFF** because it will addslash each variable even if it does not need to be addslashed. Moreover, some Web applications overlook this option, so some variables could be addslashed twice, resulting in corrupted data.

The `allow_url_include` directive is used to allow to include any file via the `require` and `include` statements, even if it does not come from your Web server. This option must be set to **OFF**, because if one application on your web server suffers of "include vulnerability", users will be able to include any file from any server and those will be executed on your own server.

PHP's Safe Mode is deprecated in the latest version of PHP, and should not be used anymore. For PrestaShop in particular, having the Safe Mode enabled can render your payment modules useless.

In short, it is **highly recommended** to have the following directives set to the indicated values:

```
register_globals = Off
magic_quotes_gpc = Off
allow_url_include = Off
safe_mode = Off
safe_mode_gid = Off
```

MySQL configuration

MySQL often has an administrator account as default ("root", "admin", ...), which gives access to all of the databases' content, no matter who the database is managed by. The administrator has all the rights, and can do every possible action. You therefore need to safekeep your databases, so as to prevent your web applications from succumbing to SQL injections (which can happen when a user succeeds in obtaining the admin password, read http://en.wikipedia.org/wiki/SQL_injection).

If you just installed MySQL, do add a password for the root account, which has no password as default.

One MySQL user per web application

Each time you install a new web application on your server, you must create a new MySQL user when just the necessary rights to handle that application's data. Do NOT use the same username to handle the databases for all of your installed web applications.

Thus, if you have access to a master MySQL account that can create other users, here's how you could do it using the command line:

```
mysql -u USERNAME -p PASSWORD
```

You could also use the following SQL query:

```
mysql> USE mysql;
mysql> CREATE USER 'username'@'servername' IDENTIFIED BY 'new_password';
```

Note that your host might give you access to an online tool to do MySQL administration tasks more easily, such as cPanel. Do use that, since you probably won't have access to the command line in that case.

Now we have a username with just enough rights to connect to the local database.

We need to allow this user to use the 'prestashop' database, and configure his rights at the same time. Here is a template for the SQL query to do that:

```
mysql> GRANT ALL ON prestashop.* TO 'new_user'@'localhost';
mysql> FLUSH PRIVILEGES;
```

We now have one user just for our 'prestashop' database. Remember to do this for each new web application you add to your server.

You can now install PrestaShop safely.

Security

Read this dedicated page, full of easy-to-apply advices.

Fine-tuning & performances

This section will help you better understand configuration variables than are not handled using the back office, but directly in configuration files.

There are four configuration files in PrestaShop, all in the `/config` folder:

- `config.inc.php`: core configuration file for PrestaShop.
- `defines.inc.php`: contains all of PrestaShop constant values. *Previously defined in `settings.inc.php`.*
- `settings.inc.php`: contains the access information to the database, as well as the PrestaShop version number.
- `smarty.config.inc.php`: contains all configuration settings pertaining to Smarty, the template/theme engine used by PrestaShop.

config.inc.php file

Most of the variables in this file are set during PrestaShop's installation, and should not be edited manually. Change this file at your own risk.

defines.inc.php file

In production mode, make sure that `define('_PS_MODE_DEV_', false);` is indeed set to `false`. In order to put PrestaShop into debug/test mode, and thus trace errors and mistake more easily, set `define('_PS_MODE_DEV_', false);` from `false` to `true`.

You can also enable the code profiling tool, which displays a lot of information at the bottom of every page: set the `define('_PS_DEBUG_PROFILING_', false);` line to `true`, then open front office or back office page. At the bottom of it, you will find a summary of the page loading performances. Note that you should really disable your store, so that visitors cannot see this information.

Among other constant values, this file contains the location for all files and folders. If you need these changed, do not forget to keep the original at hand, in case you wish to go back to the original path.

smarty.config.inc.php file

- `$smarty->caching = false;`: Smarty's cache system must be disabled because it is not compatible with PrestaShop.
- **IMPORTANT**: in production mode, `$smarty->force_compile` must be set to "false", as it will give a 30% improvement on page load time. On the other hand, when editing a `.tpl` file, you will have to delete the content of the `/tools/smarty/compile` folder (except `index.php`) in order to see the changes live. Note that this setting can also be done in the back office, in the "Advanced parameters" > "Performance" page, in the "Smarty" section.
- `$smarty->compile_check` should be left to "false".
- `$smarty->debugging` gives you access to Smarty's debugging information when your pages are displayed.

Improving PrestaShop's performances

Here are a few tips that should enable you to optimize PrestaShop.

Improving PHP performance

Whenever possible, use an opcode cache (or ask your web host to install one for you), in order to alleviate the server's processing load. Opcode means "operation code", and defines the compiled state of the dynamic files, which can then be processed faster.
PrestaShop is compatible with eAccelerator (http://eaccelerator.net/) as well as the new OPcache feature from PHP 5.5.0: http://www.php.net/manual/en/intro.opcache.php.

Improving MySQL performance

- Enable MySQL's cache (or ask your web host to do it for you), and give it a high value (for instance, 256M).
- Whenever possible, use the MySQL drop-in replacement Percona Server (http://www.percona.com/software/percona-server), which provides significant improvements over the standard MySQL server thanks to its XtraDB database engine.
  See this page for a performance comparison: http://www.percona.com/doc/percona-server/5.5/feature_comparison.html

Misc. improvements

If possible, split your static elements between different domains and sub-domains, in order to get parallel HTTP connections. To put that in place, open the `/config/defines.inc.php` file and add these lines (adapted to your needs):

```
if ( $_SERVER['REMOTE_ADDR'] != '127.0.0.1' )
{
  define( '_THEME_IMG_DIR_',   'http://img2.yourdomain.com/'       );
  define( '_THEME_CSS_DIR_',   'http://css.yourdomain.com/'        );
  define( '_THEME_JS_DIR_',    'http://js.yourdomain.com/'         );
  define( '_THEME_CAT_DIR_',   'http://img1.yourdomain.com/c/'     );
  define( '_THEME_PROD_DIR_',  'http://img1.yourdomain.com/p/'     );
  define( '_THEME_MANU_DIR_',  'http://img1.yourdomain.com/m/'     );
  define( '_PS_IMG_',          'http://img1.yourdomain.com/'       );
  define( '_PS_ADMIN_IMG_',    'http://img1.yourdomain.com/admin/' );
} else {
  define( '_THEME_IMG_DIR_',   _THEMES_DIR_ . _THEME_NAME_ . '/img/' );
  define( '_THEME_CSS_DIR_',   _THEMES_DIR_ . _THEME_NAME_ . '/css/' );
  define( '_THEME_JS_DIR_',    _THEMES_DIR_ . _THEME_NAME_ . '/js/'  );
  define( '_THEME_CAT_DIR_',   __PS_BASE_URI__ . 'img/c/'            );
  define( '_THEME_PROD_DIR_',  __PS_BASE_URI__ . 'img/p/'            );
  define( '_THEME_MANU_DIR_',  __PS_BASE_URI__ . 'img/m/'            );
  define( '_PS_IMG_',          __PS_BASE_URI__ . 'img/'              );
  define( '_PS_ADMIN_IMG_',    _PS_IMG_.'admin/'                     );
}
```

A list of tips & tricks is also available on our site:

- Tips for Optimization: http://www.prestashop.com/en/top-tips
- 10 Best Tips to speed up your PrestaShop store: http://www.prestashop.com/blog/en/10-best-tips-to-speed-up-your-prestashop-store-3/

Nginx friendly URLs

Most of the server instructions in this page pertain to the Apache web server. But some of you might prefer to rely on the Nginx web server. PrestaShop works well with Nginx, but is not able to generate the correct redirection rules for its Friendly URLs.

Here are the direction you should put in your `nginx.conf` file in order to make friendly URLs work:

```
location /PRESTASHOP_FOLDER/ {
  index /PRESTASHOP_FOLDER/index.php;

  rewrite ^/PRESTASHOP_FOLDER/api/?(.*)$ /PRESTASHOP_FOLDER/webservice/dispatcher.php?url=$1 last;
  rewrite ^/PRESTASHOP_FOLDER/([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /PRESTASHOP_FOLDER/img/p/$1/$1$2.jpg
last;
  rewrite ^/PRESTASHOP_FOLDER/([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /PRESTASHOP_FOLDER/img/p/$1/$2
/$1$2$3.jpg last;
  rewrite ^/PRESTASHOP_FOLDER/([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /PRESTASHOP_FOLDER/img
/p/$1/$2/$3/$1$2$3$4.jpg last;
  rewrite ^/PRESTASHOP_FOLDER/([0-9])([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$
/PRESTASHOP_FOLDER/img/p/$1/$2/$3/$4/$1$2$3$4$5.jpg last;
  rewrite ^/PRESTASHOP_FOLDER/([0-9])([0-9])([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$
/PRESTASHOP_FOLDER/img/p/$1/$2/$3/$4/$5/$1$2$3$4$5$6.jpg last;
  rewrite ^/PRESTASHOP_FOLDER/([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$
/PRESTASHOP_FOLDER/img/p/$1/$2/$3/$4/$5/$6/$1$2$3$4$5$6$7.jpg last;
  rewrite ^/PRESTASHOP_FOLDER/([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.
jpg$ /PRESTASHOP_FOLDER/img/p/$1/$2/$3/$4/$5/$6/$7/$1$2$3$4$5$6$7$8.jpg last;
  rewrite ^/PRESTASHOP_FOLDER/([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]
+)?/.+\.jpg$ /PRESTASHOP_FOLDER/img/p/$1/$2/$3/$4/$5/$6/$7/$8/$1$2$3$4$5$6$7$8$9.jpg last;
  rewrite ^/PRESTASHOP_FOLDER/c/([0-9]+)(-[_a-zA-Z0-9-]*)(-[0-9]+)?/.+\.jpg$ /PRESTASHOP_FOLDER/img/c/$1$2.jpg
last;
  rewrite ^/PRESTASHOP_FOLDER/c/([a-zA-Z-]+)(-[0-9]+)?/.+\.jpg$ /PRESTASHOP_FOLDER/img/c/$1.jpg last;
  rewrite ^/PRESTASHOP_FOLDER/([0-9]+)(-[_a-zA-Z0-9-]*)(-[0-9]+)?/.+\.jpg$ /PRESTASHOP_FOLDER/img/c/$1$2.jpg
last;
  try_files $uri $uri/ /PRESTASHOP_FOLDER/index.php?$args;
}
```

Note that this example uses `/PRESTASHOP_FOLDER/` as the marker for PrestaShop folder. You must replace all instances of `/PRESTASHOP_FOLDER/` by the correct path to your installation of PrestaShop.

For instance, if PrestaShop is at the root of your of your server, replace `/PRESTASHOP_FOLDER/` with simply `/` :

```
location / {
  index /index.php;

  rewrite ^/api/?(.*)$ /webservice/dispatcher.php?url=$1 last;
  rewrite ^/([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /img/p/$1/$1$2.jpg last;
  rewrite ^/([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /img/p/$1/$2/$1$2$3.jpg last;
  rewrite ^/([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /img/p/$1/$2/$3/$1$2$3$4.jpg last;
  rewrite ^/([0-9])([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /img/p/$1/$2/$3/$4/$1$2$3$4$5.jpg
last;
  rewrite ^/([0-9])([0-9])([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /img/p/$1/$2/$3/$4/$5
/$1$2$3$4$5$6.jpg last;
  rewrite ^/([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /img/p/$1/$2/$3/$4
/$5/$6/$1$2$3$4$5$6$7.jpg last;
  rewrite ^/([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /img/p/$1/$2
/$3/$4/$5/$6/$7/$1$2$3$4$5$6$7$8.jpg last;
  rewrite ^/([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /img/p
/$1/$2/$3/$4/$5/$6/$7/$8/$1$2$3$4$5$6$7$8$9.jpg last;
  rewrite ^/c/([0-9]+)(-[_a-zA-Z0-9-]*)(-[0-9]+)?/.+\.jpg$ /img/c/$1$2.jpg last;
  rewrite ^/c/([a-zA-Z-]+)(-[0-9]+)?/.+\.jpg$ /img/c/$1.jpg last;
  rewrite ^/([0-9]+)(-[_a-zA-Z0-9-]*)(-[0-9]+)?/.+\.jpg$ /img/c/$1$2.jpg last;
  try_files $uri $uri/ /index.php?$args;
}
```

If your installation of PrestaShop is using the multistore mode, you need to add a few lines for each store. For instance, if one of your stores is called "high-tech":

```
location /PRESTASHOP_FOLDER/high-tech/ {
  rewrite ^/PRESTASHOP_FOLDER/high-tech/(.*)$ /PRESTASHOP_FOLDER/$1 last;
  try_files $uri $uri/ /PRESTASHOP_FOLDER/index.php?$args;
}
```

Complete example configuration for nginx:

```
server {
    listen 80;
    #listen [::]:80;            # Uncomment this line if you also want to enable IPv6 support
    server_name example.com www.example.com;
    root /var/www/example;
    access_log /var/log/nginx/example.access.log;
    error_log /var/log/nginx/example.error.log;

    index index.php index.html; # Letting nginx know which files to try when requesting a folder


    location = /favicon.ico {
        log_not_found off;      # PrestaShop by default does not provide a favicon.ico
        access_log off;         # Disable logging to prevent excessive log sizes
    }


     location = /robots.txt {
         auth_basic off;        # Whatever happens, always let bots know about your policy
         allow all;
         log_not_found off;     # Prevent excessive log size
         access_log off;
    }

    # Deny all attempts to access hidden files such as .htaccess, .htpasswd, .DS_Store (Mac).
    location ~ /\. {
        deny all;
        access_log off;
        log_not_found off;
    }

    ##
        # Gzip Settings
        ##

        gzip on;
        gzip_disable "msie6";                                           # Do people still use Internet
Explorer 6? In that case, disable gzip and hope for the best!
        gzip_vary on;                                                   # Also compress content with other
MIME types than "text/html"
    gzip_types application/json text/css application/javascript;      # We only want to compress json, css and
js. Compressing images and such isn't worth it
        gzip_proxied any;
        gzip_comp_level 6;                                              # Set desired compression ratio,
higher is better compression, but slower
        gzip_buffers 16 8k;                                             # Gzip buffer size
    gzip_http_version 1.0;                                          # Compress every type of HTTP request

    rewrite ^/api/?(.*)$ /webservice/dispatcher.php?url=$1 last;
    rewrite ^/([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /img/p/$1/$1$2.jpg last;
    rewrite ^/([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /img/p/$1/$2/$1$2$3.jpg last;
    rewrite ^/([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /img/p/$1/$2/$3/$1$2$3$4.jpg last;
    rewrite ^/([0-9])([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /img/p/$1/$2/$3/$4/$1$2$3$4$5.
jpg last;
    rewrite ^/([0-9])([0-9])([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /img/p/$1/$2/$3/$4/$5
/$1$2$3$4$5$6.jpg last;
    rewrite ^/([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /img/p/$1/$2/$3/$4
/$5/$6/$1$2$3$4$5$6$7.jpg last;
    rewrite ^/([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /img/p/$1
/$2/$3/$4/$5/$6/$7/$1$2$3$4$5$6$7$8.jpg last;
```

```
    rewrite ^/([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])([0-9])(-[_a-zA-Z0-9-]*)?(-[0-9]+)?/.+\.jpg$ /img
/p/$1/$2/$3/$4/$5/$6/$7/$8/$1$2$3$4$5$6$7$8$9.jpg last;
    rewrite ^/c/([0-9]+)(-[_a-zA-Z0-9-]*)(-[0-9]+)?/.+\.jpg$ /img/c/$1$2.jpg last;
    rewrite ^/c/([a-zA-Z-]+)(-[0-9]+)?/.+\.jpg$ /img/c/$1.jpg last;
    rewrite ^/([0-9]+)(-[_a-zA-Z0-9-]*)(-[0-9]+)?/.+\.jpg$ /img/c/$1$2.jpg last;
    try_files $uri $uri/ /index.php?$args;

    location ~ \.php$ {
        try_files $uri =404;
        fastcgi_keep_conn on;
        include /etc/nginx/fastcgi_params;
            fastcgi_pass 127.0.0.1:9000;                    # When using TCP
        #fastcgi_pass unix:/var/run/php/php-fpm.sock;   # When using unix sockets
    }
}
```

Miscellaneous

The PrestaShop file structure

The PrestaShop developers have done their best to clearly and intuitively separate the various parts of the software.

Here is how the files are organized:

- `/admin`: contains all the PrestaShop files pertaining to the back-office. When accessing this folder with your browser, you will be asked to provide proper identification, for security reasons. **Important**: you should make sure to protect that folder with a `.htaccess` or `.htpasswd` file!
- `/cache`: contains temporary folders that are generated and re-used in order to alleviate the server's load.
- `/classes`: contains all the files pertaining to PrestaShop's object model. Each file represents (and contains) a PHP class, and its methods /properties.
- `/config`: contains all of PrestaShop's configuration files. Unless asked to, you should **never** edit them, as they are directly handled by PrestaShop's installer and back-office.
- `/controllers`: contains all the files pertaining to PrestaShop controllers – as in Model-View-Controller (or MVC), the software architecture used by PrestaShop. Each file controls a specific part of PrestaShop.
- `/css`: contains all CSS files that are not attached to themes – hence, these are mostly used by the PrestaShop back-office.
- `/docs`: contains some documentation. **Note**: it should be deleted in a production environment.
- `/download`: contains your digital products, which can be downloaded: PDFs, MP3s, etc.
- `/img`: contains all of PrestaShop's default images, icons and picture files – that, those that do not belong to the theme. This is where you can find the pictures for product categories (`/c` sub-folder, those for the products (`/p` sub-folder) and those for the back-office itself (`/admin` sub-folder).
- `/install`: contains all the files related to PrestaShop's installer. You will be required to delete it after installation, in order to increase security.
- `/js`: contains all JavaScript files that are not attached to themes. Most of them belong to the back-office. This is also where you will find the jQuery framework.
- `/localization`: contains all of PrestaShop's localization files – that is, files that contain local information, such as currency, language, tax rules and tax rules groups, states and the various units in use in the chosen country (i.e., volume in liter, weight in kilograms, etc.).
- `/log`: contains the log files generated by PrestaShop at various stages, for instance during the installation process.
- `/mails`: contains all HTML and text files related to e-mails sent by PrestaShop. Each language has its specific folder, where you can manually edit their content if you wish.
- `/modules`: contains all of PrestaShop's modules, each in its own folder. If you wish to definitely remove a module, first uninstall it from the back-office, then only can you delete its folder.
- `/override`: this is a special folder that appeared with PrestaShop 1.4. By using PrestaShop's regular folder/filename convention, it is possible to create files that override PrestaShop's default classes or controllers. This enables you to change PrestaShop core behavior without touching to the original files, keeping them safe for the next update.
- `/pdf`: contains all the template files (`.tpl`) pertaining to the PDF file generation (invoice, delivery slips, etc.). Change these files in order to change the look of the PDF files that PrestaShop generates.
- `/themes`: contains all the currently-installed themes, each in its own folder.
- `/tools`: contains external tools that were integrated into PrestaShop. For instance, this were you'll find Smarty (template/theme engine), FPDF (PDF file generator), Swift (mail sender), PEAR XML Parser (PHP tool).
- `/translations`: contains a sub-folder for each available language. However, if you wish to change the translation, you must do so using the PrestaShop internal tool, and **not** edit them directly in this folder.
- `/upload`: contains the files that would be uploaded by clients for customizable products (for instance, a picture that a client wants printed on a mug).
- `/webservice`: contains files that enable third-party applications to access PrestaShop through its API.

Moving PrestaShop

A PrestaShop installation does seldom remain at the same physical place. There are many reasons why you would need to move your PrestaShop files and data around:

- Moving your shop from your local computer to your online server.

- Moving your shop from a test sub-domain to the main domain.
- Moving your shop from one server to another.
- Moving your shop from one domain name to another.

In all of these circumstances, you must be careful to properly move both all of your files (including the custom images, your themes, the modules you bought...) and all your data (which is contained in your MySQL database).

Moving PrestaShop to a New Server

Here are the main steps when changing servers, or copying from your local hard-drive to your online server:

1. Put your shop in maintenance mode, so as to not lose new customers or orders while moving the data.
   Go to your back office, and under the "Preference" menu, open the "Maintenance" page and set the "Enable shop" option to "No".
2. Move your files:
   a. **Make a backup of all the files**: connect to your FTP server, and copy all the files and folders to your local hard-drive.
   b. **Transfer your files to your new host**: Connect to the FTP server for your new host, and copy all the files and folders that you just downloaded to your local hard-drive, as is.
3. Move your data:
   a. **Make a backup of you database (a "dump")**: connect to phpMyAdmin, click on the "Export" tab, select the database of your PrestaShop installation, and click the "Go" button. Save the downloaded file on your hard-drive. If phpMyAdmin times out before it is able to export all your data, contact your host.
   b. **Transfer the SQL dump to the new database**: connect to the new server's phpMyAdmin, click on the "Import" tab, click the "Browse..." button, find the SQL file you just downloaded, and click the "Go" button to upload it. If phpMyAdmin times out before it is able to import all your data, contact your new host.
4. Configure your shop:
   a. On the new server, open the `/config/settings.inc.php`file and update the settings for the new database server (with your own settings instead of the examples here):
      - `define('_DB_SERVER_', 'sql.domainname.com');`
      - `define('_DB_NAME_', 'prestashop');`
      - `define('_DB_USER_', 'PS-user');`
      - `define('_DB_PASSWD_', 'djsf15');`
      - `define('_DB_PREFIX_', 'ps_');`
   b. Log in to your back office, go to the "Preferences" menu, select the "SEO & URLs" page, and in the "Set shop URL" section change the domain name to your new domain. Do the same for the SSL domain.
      In effect, this will update the "ps_shop_url" SQL table (as well as the "PS_SHOP_DOMAIN" and "PS_SHOP_DOMAIN_SSL" rows in the "ps_configuration" SQL table, for retrocompatibility reasons).
5. Connect to your new FTP server and delete everything except the `index.php`files in the following folders:
   - `/cache/smarty/cache`
   - `/cache/smarty/compile`
6. Go to your back office, and in the "Maintenance" preference page, set the "Enable shop" option back to "Yes".

You should be good to go! Check that all the links are functioning, that all your products, images, modules and themes are still there, and try to create a new account and place an order so as to make sure your shop is working as expected.

Moving PrestaShop to a New Domain

Here are the main steps when moving PrestaShop to a new domain within the same server. These are mostly a simpler version of the above steps – we do not touch the data, which stays on the same MySQL server.

1. Put your shop in maintenance mode, so as to not lose new customers or orders will moving the data.
   Go to your back-office, and under the "Preference" menu, open the "Maintenance" page and set the "Enable shop" option to "No".
2. Move your files
   a. **Make a backup of all the files**: connect to your FTP server, and copy all the files and folders to your local hard-drive.
   b. **Transfer your files to your new host**: Connect to the FTP server for your new host, and copy all the files and folders that you just downloaded to your local hard-drive, as is.
3. Configuration
   a. On the new server, open the `/config/settings.inc.php` file and update the settings for the new database server (with your own settings instead of the examples here):
      - `define('_DB_SERVER_', 'sql.domainname.com');`
      - `define('_DB_NAME_', 'prestashop');`
      - `define('_DB_USER_', 'PS-user');`
      - `define('_DB_PASSWD_', 'djsf15');`
      - `define('_DB_PREFIX_', 'ps_');`
   b. Log in to your back office, go to the "Preferences" menu, select the "SEO & URLs" page, and in the "Set shop URL" section change the domain name to your new domain. Do the same for the SSL domain.
      In effect, this will update the "PS_SHOP_DOMAIN" and "PS_SHOP_DOMAIN_SSL" rows in the "ps_configuration" SQL table.
4. Connect to your new FTP server and delete everything except the `index.php`files in the following folders:

    - `/tools/smarty/cache`
    - `/tools/smarty/compile`
    - `/tools/smarty_v2/cache`
    - `/tools/smarty_v2/compile`
5. Go to your back office, and in the "Maintenance" preference page, set the "Enable shop" option back to "Yes".

You should be good to go! Check that all the links are functioning, that all your products, images, modules and themes are still there, and try to create a new account and place an order so as to make sure your shop is working as expected.