Laying the Theme's Foundations

Laying the Theme's Foundations

The first two chapters of this guide were theoretical; this one gets practical.

We are going to create the fundamental files and folders for your theme, and there are two ways of achieving that:

- Duplicate the default theme and adapt the files.
- Create every file from scratch.

Needless to say, we strongly advise you to choose the first option: a minimal PrestaShop requires many files and lines of code, some of which you are bound to forget when starting from scratch.

Building a PrestaShop is a complex endeavor. For instance, in comparison with a WordPress theme, which could work with a single `index.php` file, a minimal PrestaShop requires many more pages: home page, product page, user account pages, cart page, order process pages, etc. Building a theme for an e-commerce site implies a much more complex and intertwined set of pages and templates.

This is why we advise to start your own theme by using the foundations laid by the default theme. Complete and proven, PrestaShop's default theme ensures that all necessary pages are already in places, leaving you the freedom to rework the page display, to use your own images, to enhance it with your own scripts.

Standing on the shoulders of giants: copying the default theme

Duplicating the files of the default theme is easy, and even that can be done using two different ways.

From the back office

PrestaShop's back office can help you create a new theme folder based on any other installed theme, all in a couple of clicks:

1. Go the "Themes" preference page.
2. Click on the "Add new theme" button, at the top right of the screen.
3. In the "Import theme" screen that opens, scroll down to the "Create new theme" section and click on the "Create new theme" button.
4. In the creation form that appears, fill-in the various fields:
   - **Name of the theme**. Give it your own unique name – you can prefix it with your company's name or initials, for instance. Make sure to check on the Addons website that no other theme is already using that name, even more so if you plan to eventually sell that theme online.
   - **Preview image for the theme**. Since you do not yet have the final design for your theme, this can be done later.
   - **Default left column** / **Default right column**. Do you plan for your theme to have a sidebar of content? If so, on which side of the page? Check the options accordingly.
   - **Number of products per page**. Another informational field which you can edit later on.

- **Name of the theme's directory**. You should use the same name as your theme, in lowercase and with spaces replaced by hyphens. For instance, "My Test Theme" would yield "my-test-theme" as a folder name.
- **Copy missing files from existing theme**. This is where you indicate the theme to copy files from. This is very important! You can choose the default theme, or any other available which you want to build your theme from.
- **Responsive**. Do you plan on your theme's design to adapt to any screen size? If so, switch this option to "Yes".

5. Click on the "Save" button.

PrestaShop will create the theme's folder and copy all the needed files from the source theme, leaving you free to experiment with those files.

From your operating system / FTP server

You can of course create that copy yourself:

1. Go to the `/themes` folder for your installation of PrestaShop (either online or on your desktop).
2. Create a new folder for your theme. It should be the final name of your theme, in a single lowercase word. Make sure to check on the Addons website that no other theme is already using that name, even more so if you plan to eventually sell that theme online.
3. Copy the content of source theme's folder (for the 1.6 default theme, `/default-bootstrap`) and paste it in the newly created folder.

That's it!

Cleaning up

Both the default theme and your installation of PrestaShop contain a payload of content and styles that are not necessarily useful to your own theme. For instance, many modules are installed and activated by default by PrestaShop's installer. While some are necessary for the proper functioning of a complete store, others can simply be put aside while you build the theme.
It is your duty as a theme developer to build templates and styles for at least all the default PrestaShop modules (or at least adapt the default ones), along with the ones for any additional modules that you plan your theme to support.

The point here is that a theme must come packed with custom templates and CSS files for the default PrestaShop modules. These template files are stored in the `/modules` subfolder of the theme's folder, while the styles are in the `/css/modules` subfolder.
For instance, in the default theme, the files for the Layered Navigation module can be found in the following paths:

- Template file: `/themes/default-bootstrap/modules/blocklayered/blocklayered.tpl` .
- CSS file: `/themes/default-bootstrap/css/modules/blocklayered/blocklayered.css` .

As you can see, if all you want to change in a module's front office appearance while keeping its organzation, you only have to edit its CSS file and leave its template file alone. For instance, to change the styling of the Layered Navigation module, you should put your customized version in this folder: `/themes/YOUR_THEME/css/modules/blocklayered/blocklayered.css`. Just make sure to use the same file path as the original module files.

Necessary modules

The necessary module templates are:

Module name	Why it is necessary
blockcart	Displays the whole order & payment process.
blockmyaccount	Displays the user creation process.

You simply cannot sell a product if your theme does not support these modules.

Must-have modules

There also are modules which, while not necessary for a functioning store, should still be included when designing a theme. You should try your best to build your theme with these modules in mind.

The "Must have" module templates are:

Module name	Why it is necessary
blockcategories	Displays the product categories.
blockcms	Lists and displays the CMS pages (i.e. Terms & Conditions, Legal notice, etc.).
blockcontact	Displays the Customer Service information.
blockcontactinfos	Displays the stores contact info.
blockmyaccountfooter	Displays links to the user's account pages in the footer.
blocksearch	Displays the search engine and its results.
blocktags	Displays the product tags.
homefeatured	Displays featured products.

Good-to-have modules

Finally, these modules are not as important as the others ones, but bring a lot of value to your store, and helps your customers discover products and learn more about your store. Again, you should design your store with these modules activated.

The "Good to have" module templates are:

Module name	Why it is necessary
blockbestsellers	Displays the best-selling product.
blocklayered	Displays layered navigation filters.
blocklinks	Displays additional custom links.
blockmanufacturer	Lists and displays the manufacturers/brands of the store's products.
blocknewproducts	Displays the newest products.
blocknewsletter	Displays a form where customers can subscribe to your store's newsletter.
blockrss	Displays the content of an RSS feed from another site.
blocksocial	Displays information about your store's social networking pages.
blockspecials	Displays the current discounts.
blocstore	Displays a link to the store located.
blocksupplier	Lists and displays the suppliers of the store's products.
blockviewed	Lists the products that the customer viewed last.
blockwishlist	Displays the customer's wishlists.

productcomments	Displays a comment section in each product page.

All these module templates are included by default in the default theme's `/modules` folder, because they are front-end features that are needed by that theme. You can safely disable/uninstall any other module in the back office "Modules" page. This enables you to start on a somewhat clean slate.

> A fully clean slate would be to disable all modules and re-install them one by one, enabling you to integrate them into your design while building you theme. This is a good way to work, as you it helps you know which content broke your page layout, but it takes longer to reach your goal. Keep a known set of essential modules helps you build your theme faster while making sure it will work in most configuration.

Creating content

Your theme will display content taken from the PrestaShop database. Whether you plan on keeping the theme to yourself or share/sell it for others to use, you simply cannot start designing it without content, along with the activation of some key features that any store might use, along with yourself.

The demo data installed with PrestaShop is enough to help with it, as it features products, categories, stores, etc. Starting with a fresh installation of PrestaShop gives you a head-start with demo content, while empty stores will require you to start adding content (either fake or real) to the store in order to actually see your theme react to it.

Design!

Now that the default theme has been turned into a folder of its own, it is time for you to explore its files: Smarty templates, CSS rules, JavaScript codes, location of the hooks and content blocks... Everything can be changed, and it is up to you to rework it the way you want!

The CSS files

You can edit your theme's styles by editing its CSS files.

The `global.css` file contains the global structure and the most important part of the design for the theme.

Usually, there is one stylesheet per controller. For instance, the product's page has a `product.css` file.

If you use Sass/Compass, you must edit the SCSS files first, then regenerate the CSS files.
Once the stylesheet is generated from a SCSS file, the SCSS line and file are referenced within the file.

Sample code from product.css

```
/* line 6, ../sass/product.scss */
.primary_block {
  margin-bottom: 40px;
}
/* line 9, ../sass/product.scss */
.top-hr {
  background: #c4c4c4;
  height: 5px;
  margin: 2px 0 31px;
}
```

```
.primary_block {
  margin-bottom: 40px;
}
.top-hr{
  background: $top-line-color;
  height: 5px;
  margin: 2px 0 31px;
}
```

The images

Images which are used by the theme (not by products) are to be stored in the theme's `/img` folder, while image which are to be used by the shop itself are to be stored in the `/img` folder at the root of your PrestaShop installation.

Icons are a special case: PrestaShop 1.6 uses FontAwesome as its icon set: http://fontawesome.io/. It is a font made of icons, giving you scalable vector icons that can instantly be customized.

Using FontAwesome enables you to:

- use a single file to display many different icons.
- enjoy great flexibility: size, color, drop shadow, and anything that can be done with the power of CSS.
- have an excellent render on all screen sizes : PC, TV, Retina, etc.)

The template files

A template files (`.tpl`) is Smarty's way to separate the content from the way that content is presented. The template only have a few dynamic elements (where your content goes). This facilitates the design and update of your sites, both for your content and its presentation.

Template file for the HTTP 404 Error page

```
<div class="pagenotfound">
    <div class="img-404">
        <img src="{$img_dir}/img-404.jpg" alt="{l s='Page not found'}" />
    </div>
    <h1>{l s='This page is not available'}</h1>
    <p>
        {l s='We\'re sorry, but the Web address you\'ve entered is no longer available.'}
    </p>
    <h3>{l s='To find a product, please type its name in the field below.'}</h3>
    <form action="{$link->getPageLink('search')|escape:'html':'UTF-8'}" method="post" class="std">
        <fieldset>
            <div>
                <label for="search_query">{l s='Search our product catalog:'}</label>
                <input id="search_query" name="search_query" type="text" class="form-control grey" />
                <button type="submit" name="Submit" value="OK" class="btn btn-default button button-small">
                    <span>{l s='Ok'}</span>
                </button>
            </div>
        </fieldset>
    </form>
    <div class="buttons">
        <a class="btn btn-default button button-medium" href="{$base_dir}" title="{l s='Home'}">
            <span><i class="icon-chevron-left left"></i>{l s='Home page'}</span>
        </a>
    </div>
</div>
```

The template engine uses {...} to handle instructions. The rest of the document is sent to the browser as is.

It is possible to use a template to generate HTML files, and also XML files, text files, etc.

The PDF files

The PDF files are also generated from Smarty templates (`.tpl` files). The main difference with the template files used to generate HTML, is that the PDF templates do not allow for external resources such as CSS. Therefore you must use internal or inline styling in order to change the style of your invoice, order slip, return slip, etc. A default PrestaShop installation comes with two style templates: `delivery-slip.style-tab.tpl` and `invoice.style-tab.tpl`. You can find these in the folder `/pdf/`. The following table shows how the style tabs are linked:

PDF Template	Style tab
Delivery slip	`delivery-slip.style-tab.tpl`
Supply order	`invoice.style-tab.tpl`
Order return	`invoice.style-tab.tpl`
Invoice	`invoice.style-tab.tpl`
Order slip	`invoice.style-tab.tpl`

Note that if you would like to apply a different style template, you will have to override the corresponding PDF Class in the directory `/classes/pdf/`.

The internal styling in the style tabs can only be applied to the main PDF template files, such as `invoice.tpl`, `delivery-slip.tpl`, etc. In order to reuse the variables in this template in templates which are included, you can assign Smarty variables or apply inline styling separately in each of these files.

Scratching that itch: creating all files from zero

Oh wow. Really? You want to do it all by hand? That's courageous, but we'll try to help.

First, here is the list of necessary template files (spoiler alert: there are 60 of them. Yes, all are necessary to various PrestaShop features):

File name	Why it is necessary	Other template files used by this template in the default theme
404.tpl	Displays when a file cannot be found.	
address.tpl	Enables the customer to create a new address.	• errors.tpl
addresses.tpl	Enables the customer to view her current addresses.	
authentication.tpl	Enables the customer to log into her account.	• order-steps.tpl • errors.tpl
best-sales.tpl	Displays the best-selling products.	• product-sort.tpl • nbr-product-page.tpl • product-compare.tpl • pagination.tpl • product-list.tpl
breadcrumb.tpl	Displays the category path to the current product/category.	• breadcrumb.tpl

category-cms-tree-branch.tpl	Runs through the CMS categories in order to display them.	• category-cms-tree-branch.tpl
category-count.tpl	Displays the number of products in a category.	
category-tree-branch.tpl	Runs through the product categories in order to display them.	• category-tree-branch.tpl
category.tpl	Displays the content of a category: scene, image, text, product comparator, etc.	• errors.tpl • scenes.tpl • category-count.tpl • product-sort.tpl • nbr-product-page.tpl • product-compare.tpl • pagination.tpl • product-list.tpl • product-compare.tpl
cms.tpl	Displays the content of a CMS page.	
contact-form.tpl	Displays the customer contact form.	• errors.tpl
discount.tpl	Displays the list of the customer's vouchers.	
errors.tpl	Displays the current error(s).	
footer.tpl	Displays the footer.	• global.tpl
global.tpl	Defines several Smarty variables, most notable JavaScript ones.	
guest-tracking.tpl	Displays the tracking page for guest customers (visitors with no account).	• order-detail.tpl
header.tpl	Displays the header: HTML doctype, links to CSS files, etc.	• breadcrumb.tpl
history.tpl	Displays all her previous orders to the customer.	• errors.tpl
identity.tpl	Displays and updates the customer's personal information.	• errors.tpl
layout.tpl	Calls upon the main bricks of the theme: header, footer, columns, current template and Live Edit.	• header.tpl • footer.tpl
maintenance.tpl	Displays a special page for when the store is in maintenance.	
manufacturer-list.tpl	Displays a list of all manufacturers.	• errors.tpl • nbr-product-page.tpl • pagination.tpl
manufacturer.tpl	Display the products from a single manufacturer.	• errors.tpl • product-sort.tpl • nbr-product-page.tpl • product-compare.tpl • pagination.tpl • product-list.tpl
my-account.tpl	Displays the customer's account page.	
nbr-product-page.tpl	Displays the number of products in the current page.	
new-products.tpl	Displays a block with the new products.	• product-sort.tpl

		• nbr-product-page.tpl • product-compare.tpl • pagination.tpl • product-compare.tpl • pagination.tpl
order-address-multishipping-products.tpl	Displays the addresses to deliver a product to in a multishipping situation.	• order-address-product-line.tpl
order-address-multishipping.tpl		• order-steps.tpl • errors.tpl • order-address-multishipping-products.tpl
order-address-product-line.tpl		
order-address.tpl		• order-steps.tpl • errors.tpl
order-carrier.tpl		• order-steps.tpl • errors.tpl
order-confirmation.tpl		• errors.tpl • order-steps.tpl
order-detail.tpl		
order-follow.tpl		
order-opc-new-account.tpl		
order-opc.tpl		• shopping-cart.tpl • order-address.tpl • order-opc-new-account.tpl • order-carrier.tpl • order-payment.tpl • errors.tpl
order-payment.tpl		• errors.tpl • order-steps.tpl • shopping-cart-product-line.tpl
order-return.tpl		• errors.tpl
order-slip.tpl		
order-steps.tpl		
pagination.tpl		
password.tpl		• errors.tpl
prices-drop.tpl		• product-sort.tpl • nbr-product-page.tpl • product-compare.tpl • pagination.tpl • product-list.tpl • product-compare.tpl • pagination.tpl
product-compare.tpl		

product-list-colors.tpl		
product-list.tpl		
product-sort.tpl		
product.tpl		• errors.tpl • product-list.tpl
products-comparison.tpl		
restricted-country.tpl		
scenes.tpl		
search.tpl		• errors.tpl • product-sort.tpl • nbr-product-page.tpl • product-compare.tpl • pagination.tpl • product-list.tpl
shopping-cart-product-line.tpl		
shopping-cart.tpl		• order-steps.tpl • errors.tpl • shopping-cart-product-line.tpl
sitemap.tpl		• category-tree-branch.tpl • category-cms-tree-branch.tpl
stores.tpl		
store_infos.tpl		
supplier-list.tpl		
supplier.tpl	Makes it possible to display the list of products per supplier.	• errors.tpl • product-sort.tpl • nbr-product-page.tpl • product-compare.tpl • pagination.tpl • product-list.tpl