# Magento for Designers: Part 3

Magento is a stunningly powerful e-commerce platform. In this miniseries, we’ll learn how to get started with the platform, getting to know the terminologies, setting up a store and all related aspects of it and finally learn how to customize it to make it our very own.

In this third part, we’ll focus on the process behind theming Magento: how to install themes, the various concepts you’ll need to understand to create a theme and the general file structure. Excited? Let’s get started!

## The Full Series

* [Part 1: Installation and Setup](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-i/)
* [Part 2: Products, Taxes, Categories, Payment Gateways, etc.](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-2/)
* [Part 3: Theming](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-3/)
* [Part 4: Building the Theme](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-4/)
* [Part 5: Building the Product Information Page](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-5/)

## A Quick Recap

In the last part, we saw how to get your Magento store from installed to ready for deployment including how to set up your products, product categories, taxes, shipping, payment gateways and many more.

Today, we’ll look at the basics of Magento theming. We’ll learn the general idea behind Magento themes, the various terminologies behind it and the basic structure of a theme.

## Magento Theme Basics

First up, theming Magento isn’t really as hard as it is purported. It’s a little different from how WordPress or Joomla handles themes, yes, but definitely not difficult. All you need to know is a little know how to start theming like a pro!

To make it brutally simple, a Magento theme is a collection of PHTML, CSS and JS files thrown in together along with XML files to define the structure. A PHTML file consists of regular HTML markup interspersed by PHP code for the functionality. In case, you’re confused, a random block of code looks like so:

[view plaincopy to clipboardprint?](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-3/)

1. <div class="quick-access">
2. <?php echo $this->getChildHtml('store\_language') ?>
3. <p class="welcome-msg"><?php echo $this->getWelcome()?></p>
4. <?php echo $this->getChildHtml('topLinks') ?>
5. </div>

See? It’s really simple once you wrap your head around it. If you’ve worked with creating themes for other systems, great, you’ll pick this up rather quickly. If not, no worries, I’ll walk you through the entire process.

Note that in Magento, the front end and the back end are skinned completely separately. I’m assuming most of you won’t need to skin the backend so I’ll stick to theming the front end alone.

## Installing a Theme

Before we start, a number of people DMed me through Twitter/emailed me asking the same question: how to install a theme. I’ll talk about it first.

There are two ways to install a Magento theme:

* The traditional method where you can just copy the packaged theme to appropriate folder
* Magento Connect

I’ll talk briefly about both.

### Direct Upload/Copy

The first method is the one you’re used to. Download a theme, upload it and done. But you’ll need to know where to upload since this works a little differently than you’d assume.

Themes are packaged differently according to the source but at it’s core, you have 2 folders:

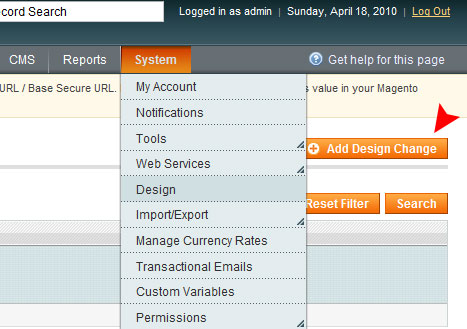
* app
* skin

You can just drag these to the root of the installation and let it merge with the existing data.

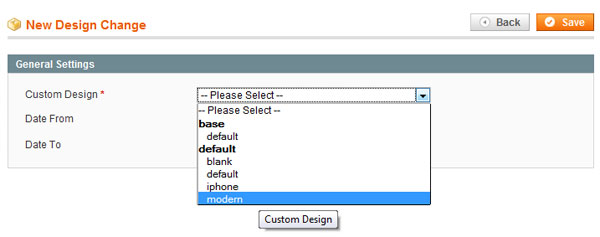
If by chance, you get the theme packaged as a collection of 3 folders, don’t worry.

The folder containing the PHTML files and the one containing the XML files go into root/app/design/frontend/default/themename while the one containing the CSS files, images and other assets goes into root/skin/frontend/default/themename.

Right now, this is all you need to do. I’ll explain why each part goes to a specific location later below. You can activate your theme now.



Navigate to System -> Design and click on Add Design Change.

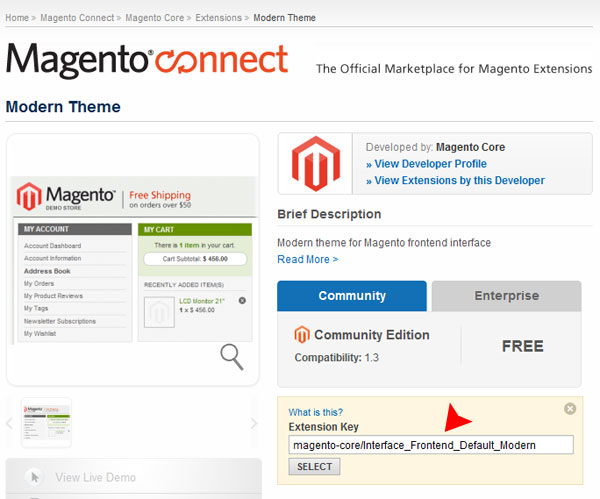


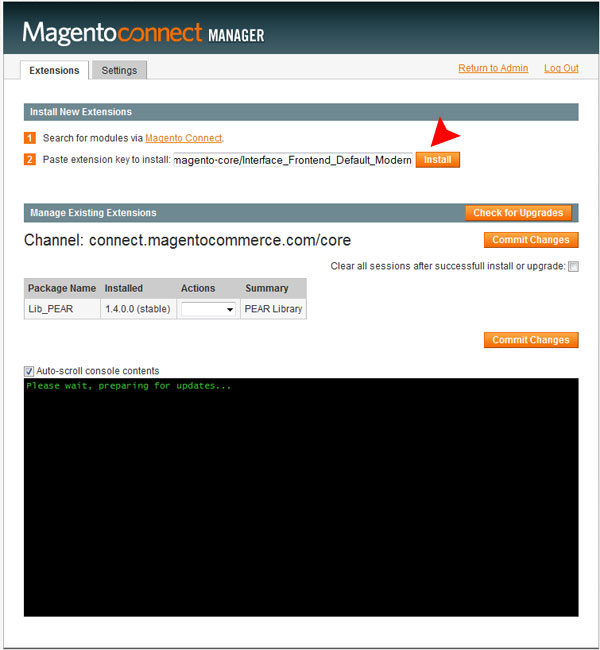
Choose the theme you want, click on save and you’re done.

### Magento Connect

Using Magento Connect is easier provided it is available there. Navigate to System ->Magento Connect -> Magento Connect Manager.

After logging in, you’ll be asked to enter the extension key of the theme you want to install. Enter the key and wait for the system to do it’s thing.





After it has downloaded the necessary files and placed them where they need to be, you can now activate the theme like before.

## Magento Design Concepts You Need to Master

When working with Magento, there are a few design related concepts you need to wrap your mind around before you can even start modifying the default theme.

### Layouts

Layouts is a clever, new idea in Magento. This system lets you define a page’s, any page’s, structure through properly formed XML tags.

Essentially, you can dictate which section of the page goes where by changing just a few attributes in an XML file. Each view or module gets it’s layout specified by its own XML file.

Layouts in Magento is a big topic and just a few paragraphs here won’t do it justice. Along the way, I’ll cover all the necessary information you need to build your own theme along with a detailed article on layouts to cover all the advanced things you can do with this functionality.

For now, if you’re interested, here is a small snippet to get an idea of what layouts are:

[view plaincopy to clipboardprint?](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-3/)

1. <block type="page/html\_notices" name="global\_notices" as="global\_notices" template="page/html/notices.phtml" />
3. <block type="page/html\_header" name="header" as="header">
4. <block type="page/template\_links" name="top.links" as="topLinks"/>
5. <block type="page/switch" name="store\_language" as="store\_language" template="page/switch/languages.phtml"/>
6. <block type="core/text\_list" name="top.menu" as="topMenu"/>
7. <block type="page/html\_wrapper" name="top.container" as="topContainer" translate="label">
8. <label>Page Header</label>
9. <action method="setElementClass"><value>top-container</value></action>
10. </block>
11. </block>
13. <block type="page/html\_breadcrumbs" name="breadcrumbs" as="breadcrumbs"/>
15. <block type="core/text\_list" name="left" as="left" translate="label">
16. <label>Left Column</label>
17. </block>
19. <block type="core/messages" name="global\_messages" as="global\_messages"/>
20. <block type="core/messages" name="messages" as="messages"/>

### Templates

Templates consist of PHTML files filled with regular HTML markup and PHP code. Similar to WordPress, you use a number of predefined methods to specify the output. Just like with other popular systems, important sections like the header, footer and the sidebar are placed in separate files and pulled in when necessary.

You can have different templates for each view of Magento. For example, you can have different code for a wish list or a checkout page instead of using the same look for the entire site.

Here is a piece of a template for the curious:

[view plaincopy to clipboardprint?](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-3/)

1. <ul class="products-grid">
3. <li class="item">
4. <p class="product-image">
5. <a href="<?php echo $\_product->getProductUrl() ?>"
6. title="<?php echo $this->htmlEscape($this->getImageLabel($\_product, 'small\_image')) ?>">
7. <img src="<?php echo $this->helper('catalog/image')
8. ->init($\_product, 'small\_image')
9. ->resize(100, 100); ?>"
10. width="100" height="100"
11. alt="<?php echo $this->htmlEscape($this
12. ->getImageLabel($\_product, 'small\_image')) ?>"
13. title="<?php echo $this->htmlEscape($this
14. ->getImageLabel($\_product, 'small\_image')) ?>" />
15. </a>
16. </p>
17. <h5 class="product-name">
18. <a href="<?php echo $\_product->getProductUrl() ?>"
19. title="<?php echo $this->htmlEscape($\_product->getName()) ?>">
20. <?php echo $this->htmlEscape($\_product->getName()) ?></a>
21. </h5>
22. <?php echo $this->getReviewsSummaryHtml($\_product, 'short') ?>
23. <?php echo $this->getPriceHtml($\_product, true, '-new') ?>
25. </li>
26. <?php if ($i%$\_columnCount==0 || $i==count($\_products)): ?>
27. </ul>

Looks a little messy, I know but strip out the PHP parts and you’ll see how similar it is to other systems.

### Skins

Skins are nothing but the CSS files, JavaScript files, images and other assets you’re using in the markup to create your design. Essentially all non PHP assets go here. Fonts for embedding? Some swanky flash demo? A spiffy piece of SVG? All of those fall under this category.

### Blocks

Blocks are the integral building blocks of a theme and let you build your theme in a modular fashion.

As part of layouts, this forms the backbone of Magento’s strong templating system. Blocks are essentially sections which you can move around using the XML mentioned above to modify how a page is presented.

Blocks need to reference a relevant template file so that Magento can pull in the required file. A little confused? Here is an example.

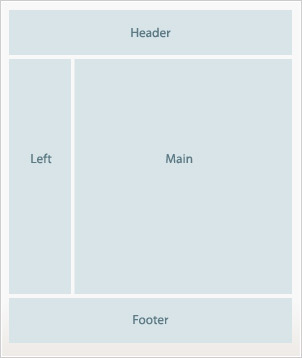
[view plaincopy to clipboardprint?](http://net.tutsplus.com/tutorials/php/magento-for-designers-part-3/)

1. <block type="page/html\_breadcrumbs" name="breadcrumbs" as="breadcrumbs"/>

We essentially define a new block, which template to load by specifying the type of block and a name. It’s a little different from what we’ve been used to but trust me you’ll get it once you get started developing. Either way, I’ll cover blocks a bit more in detail when we’re building our theme and still more I’ll do a full write up on layouts and blocks down the line so don’t worry if it doesn’t make complete sense now. Just get a general feel for the topics at hand.

### Structural Blocks

A structural block defines the basic structure of a page. Think HTML 5 header, footer and aside sections. They were created for the sole purpose of visual demarcation of a design.

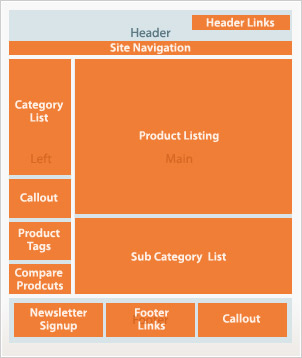


From the Magento docs

### Content Blocks

Content blocks are similar to your regular container/wrapper DIV elements you use in a design. Just like with design, each content block contains a specific functionality or purpose. A menu in your header, a callout in the sidebar, legal clarifications in the footer all go into separate content blocks.

Remember, content blocks are still blocks and map to a specific PHTML file to generate and render its HTML contents.



From the Magento docs

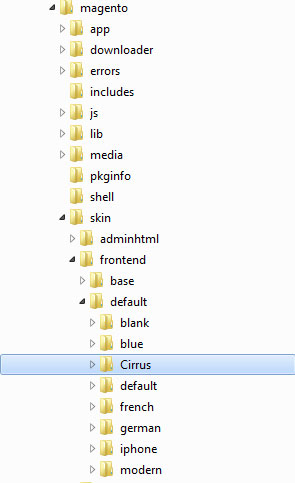
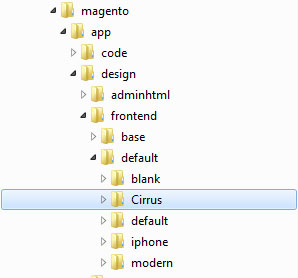
### Interface

Mentioned finally because from a strict theming perspective of a beginner, this shouldn’t come into play for quite a while.

To be simple, an interface is a named collection of themes you can leverage to define the look of your store.

## Important Locations to Keep in Mind Whilst Theming

Just like other powerful software, Magento has a complex file structure. However, for theming along, you can narrow your focus down considerably.



Here are the locations you’ll be working on when creating a theme:

* root/app/design/frontend/default – The folder of the default interface. Aptly named default, by default. (Heh!)
* root/app/design/frontend/default/Cirrus – The folder for the theme we will be building. I’ve named our theme, Cirrus
* root/skin/frontend/default – The folder of the default interface.
* root/skin/frontend/default/Cirrus – The folder where all the assets for our theme will be placed.

## A Theme’s Directory Structure

Magento requires that your executable PHP content be placed seperately from your static assets which is why you have a separate skin directory on your root. While this may seem counter-productive at first, once you’ve slightly adapted your workflow, you’ll realize that this move increases the general security of your installation..

Nevertheless, a theme is typically split into the following parts.

* Layouts – root/app/design/frontend/default/Cirrus/layouts
* Templates – root/app/design/frontend/default/Cirrus/templates
* Skins – root/skin/frontend/default/Cirrus

## The [Second to] Last Word

And we are done! We looked at the basic concepts behind theming Magento and managing themes. Hopefully this has been useful to you and you found it interesting. Since this is a rather new topic for a lot of readers I’ll be closely watching the comments section so chime in there if you’re having any doubts.

Questions? Nice things to say? Criticisms? Hit the comments section and leave me a comment. Happy coding!

## What We’ll Build in the Upcoming Parts

So far, we’ve been dealing strictly theoretically with the platform. A necessity considering Magento’s size and scope. But now that we have all the basics nailed down we can move on to the fun part.

Remember how when creating a skin for a CMS/generic system you always start from a skeleton and build outwards? Like Kubrick for WordPress? If you thought we were going to take one and start building a theme out of it, you thought wrong. No, sir. We’re going to build a custom, bare bones skin similar to the Blank skin completely from scratch. A skin you can use yourselves as a base for your own skin.

All this and more in the upcoming parts. Stay tuned!