Magento's Theme Hierarchy - Changes in CEv1.4 and EEv1.8

**Category:** [Themes and Design](http://www.magentocommerce.com/knowledge-base/categories/category/themes-and-design)   
*Kara Heinrichs, Magento*

**Applicable to:** CEv1.4+, EEv1.8+

Magento’s theme architecture was changed between Community Edition (CE) v1.3 and CE v1.4 and between Enterprise Edition (EE) v1.7 and EE v1.8 to make custom themes easier to maintain and more “upgrade proof.” In these releases the theme files were also refactored and changed considerably to improve performance and accessibility, but the biggest change was to the overall theme structure and hierarchy used by Magento.

This article describes the new Magento theme structure and hierarchy and how it impacts custom theme development. It’s intended for designers and developers who worked with Magento themes in CE v1.3 and EE v1.7 and before and who need to understand the changes.

**In this article...**

* [Understanding the Base Package](http://www.magentocommerce.com/knowledge-base/entry/magentos-theme-hierarchy#1)
* [The Design Fall-Back Hierarchy](http://www.magentocommerce.com/knowledge-base/entry/magentos-theme-hierarchy#2)
* [Frequently Asked Questions](http://www.magentocommerce.com/knowledge-base/entry/magentos-theme-hierarchy#3)
  + [How Do I Make a Custom Package Live on the Frontend?](http://www.magentocommerce.com/knowledge-base/entry/magentos-theme-hierarchy#3.1)
  + [What About default/default?](http://www.magentocommerce.com/knowledge-base/entry/magentos-theme-hierarchy#3.2)
  + [Is It Ever OK to Edit base/default?](http://www.magentocommerce.com/knowledge-base/entry/magentos-theme-hierarchy#3.3)
* [Where to Next?](http://www.magentocommerce.com/knowledge-base/entry/magentos-theme-hierarchy#4)

Understanding the Base Package

Introduced in CE v1.4 and EE v1.8 was the “base” package and a change to Magento’s design fall-back hierarchy that makes the base/default theme a true cross-package fallback option. The role of the base package is to provide hooks to all of Magento’s core functionality, so that custom themes can include in them just the changes to that core functionality that are specific to the design or business they’re intended to support. This provides a cleaner codebase and a better upgrade path for your themes.

In the directory structure, the base package is located at:

* app/design/frontend/base/default/ - contains all of the layout and template files necessary to support core Magento functionality.
* skin/frontend/base/default/ - contains some CSS and Javascript files, which support core functionality. It does not, however, contain all of the CSS and image files necessary to style a site, because those are specific to a design and not to the core.

Base isn’t really a full theme *per se*, since it lacks most of the skin files—CSS, images, etc. It’s a repository for making core Magento functionality available to the frontend. So don’t make any of your customizations there--files in the base package should not be edited. Custom themes should be made inside of their own design package directories. Files that need to be changed can be copied from base to your custom design package and edited there or you can create a set local files to hold your mods.

* **Rule 1:** Do NOT edit the files in the base package
* **Rule 2:** Do not create a custom theme inside of the base package

The Design Fall-Back Hierarchy

Magento has always used fall-back logic in rendering themes, but with CE v1.4 and EE v1.8, this has been expanded to include the base package as the final cross-package fall-back point. (In previous Magento versions, fall back was only inside of a theme and customized frontend functionality could not be shared between packages without replicating all of the files to each package.)

An example of how Magento’s fall-back logic works is, if your custom theme calls for a CSS file called ‘styles.css’ but the Magento application is unable to find the file in your custom theme, Magento will go to the next theme in the hierarchy to find the file. Should this fail, it will continue working down the hierarchy of themes until it’s able to locate the file called ‘styles.css.’ This method of building design is called fall backs, because the application ‘falls back’ to the next possible source of required files in order to retrieve and load a requested file.

The fall-back hierarchy in Magento CE v1.4+ and EE v1.8+ is as follows.

1. Look for requested file in:
   * app/design/frontend/*custom\_package*/*custom\_theme*/
   * skin/frontend/*custom\_ package*/*custom\_theme*
2. If not found, look for requested file in:
   * app/design/frontend/*custom\_package*/default
   * skin/frontend/*custom\_package*/default
3. If not found, look for requested file in:
   * app/design/frontend/base/default
   * skin/frontend/base/default
4. If not found, a rendering error will occur.

When used effectively, this fall-back mechanism allows you to edit and maintain only the files you need to change as a custom theme and all of the other functionality is provided by either the custom package default theme or the base package.

For example, Magento maintains 4 CE demo themes—default, blank, modern, and iPhone. Before the introduction of the base package, all theme files had to be copied to each package and maintained, tested and debugged in each. With the introduction of the base package, we maintain 3 times less code. For example, the default/default and default/blank themes are implemented with just CSS changes and get all their template and layout files from the base package.

Ultimately, this will also make your custom themes more “upgrade proof” and allow you to maintain less code as well. Most custom themes only customize a subset of the default Magento functionality, so now all of the core functionality will be made available by the base package and can be overridden selectively by a custom package/theme.

* **Rule 3:** Create your customized themes inside of their own design package. Make directories atapp/design/frontend/*your\_custom\_package*/default andskin/frontend/*your\_custom\_package*/default and build your custom theme there.
* **Rule 4:** Do not copy all the files from base/default into your custom package. Copy only the files that you modify.

Frequently Asked Questions

How Do I Make a Custom Package Live on the Frontend?

This hasn’t changed at all. Go to the Admin Panel and select **System** → **Configuration** from the top menu. Click on the **Design** tab on the left-hand side. In the “Current Package Name” field, enter *your\_custom\_package*.

What About default/default?

It’s time to let default/default go. Give it a Viking funeral if that pains you and let it go. The problem withdefault/default is that it gets overwritten during any Magento upgrade. So designers and developers who would customize default/default directly would have their work overwritten every time. (base/default also gets overwritten during upgrades, which is why you should not customize those files directly either.)

Even though copying default/default to a new theme or design package directory would make sure the custom theme didn’t get overwritten, because all of the files were copied, it would also prohibit any new functionality included with each upgrade from being automatically available after an upgrade. It also brings along styling and additional files you may not need at all for your theme.

Default/default is also not available across multiple packages, so changes there would still have to be replicated to (and maintained in) all other packages in order to be available to each.

* **Rule 5:** Do not create your custom theme inside of the default/default directories, even though there are old instructions that may tell you to do so. This approach was never recommended.
* **Rule 6:** Do not copy all the files from default/default into your custom package as a starting point, even though there are old instructions that may tell you to do so. This approach has been deprecated.
* **Exception to Rule 6:** Because base/default does not contain full CSS files, you may want to copy the CSS from one of the demo themes into your new custom themes as a starting point. Many designers and developers prefer to start with nothing and build their CSS from scratch though, so you can do this too. In general, if you do copy over CSS and image files to your new custom package, the blank theme (default/blank) provides the leanest CSS.
* **2nd Exception to Rule 6:** Enterprise customers for now DO need to copy the files from enterprise/default into their custom package as a starting point. In a future release, the Enterprise files will be folded into the base/default directories but they are not in EEv1.8.

Is It Ever OK to Edit base/default?

Yes, there might be times when you want/need to edit base/default directly. But note: YOU DO SO AT YOUR OWN RISK. Keep track of your changes, you may need to manually restore them after upgrades.

The primary reason for allowing yourself or your designers and developers to edit the files in base/default is if you have modifications that you would like to make available across multiple design packages by making them part of the fall-back hierarchy. This might be the case if you have multiple brands/stores that have unique designs and business needs (and therefore need their own design packages) but share some customized functionality. Or, if you are a 3rd-party extension developer, you can put in your theme files into base/default to guarantee that extension’s portability to any custom theme.

* **Rule 7:** Break the rules if you have to, just do so knowingly.

Where to Next?

Watch for more articles in the Knowledge Base about theming in Magento. In the meanwhile, here are two good wiki articles on two different methods for creating custom themes:

* [How to Create a New Package/Theme](http://www.magentocommerce.com/wiki/4_-_themes_and_template_customization/0_-_theming_in_magento/package) - Discusses creating a custom package by moving/copying just the files you change
* [Designing for Magento](http://www.magentocommerce.com/wiki/4_-_themes_and_template_customization/0_-_theming_in_magento/designing-for-magento) - Discusses creating a custom package by using a local.xml file and localized changes

**Discussion:**