



Quick answers to common problems

# Magento 1.4

## Theming Cookbook

Over 40 recipes to create a fully functional, feature rich, customized Magento theme

**Jose Argudo Blanco**

**[PACKT]** open source\*  
PUBLISHING community experience distilled

# Magento 1.4 Theming Cookbook

Over 40 recipes to create a fully functional, feature rich, customized Magento theme

**Jose Argudo Blanco**



BIRMINGHAM - MUMBAI

# Magento 1.4 Theming Cookbook

Copyright © 2011 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: August 2011

Production Reference: 1040811

Published by Packt Publishing Ltd.  
Livery Place  
35 Livery Street  
Birmingham B3 2PB, UK.

ISBN 978-1-849514-24-8

[www.packtpub.com](http://www.packtpub.com)

Cover Image by Parag Kadam ([paragvkadam@gmail.com](mailto:paragvkadam@gmail.com))

# Credits

**Author**

Jose Argudo Blanco

**Copy Editor**

Neha Shetty

**Reviewers**

Ivan Chepurnyi

Hans Kuijpers

Fernando J. Miguel

**Project Coordinator**

Joel Goveya

**Proofreader**

Aaron Nash

**Acquisition Editor**

Dilip Venkatesh

**Indexer**

Rekha Nair

**Development Editor**

Hithesh Uchil

**Production Coordinator**

Aparna Bhagat

**Technical Editor**

Kavita Iyer

**Cover Work**

Aparna Bhagat

# About the Author

**Jose Argudo Blanco** is a web developer from Valencia, Spain. After finishing his studies, he started working for a web designing company. Then, six years later, he decided to start working as a freelancer.

Now that some years have passed since he has been working as a freelancer, he has decided to work with tools he likes. Tools such as Joomla!, CodeIgniter, Magento, jQuery, and other known open source technologies. He thinks it's the best decision he has ever taken.

In 2009 he published his first book, *CodeIgniter 1.7*, which you can also find at the Packt website. And after that he published *Joomla! 1.5 JavaScript jQuery*. Later, he started working on this Magento book.

# About the Reviewers

**Ivan Chepurnyi** has experience in web development for more than eight years and at least six of them are related to e-commerce.

He started his career as a Magento Developer when he joined Varien (now it is Magento Inc.) in May 2007 as one of the core team members. Since then, he has been working only with Magento, which is considered to be the most flexible and powerful open source e-commerce system.

Currently, he provides training on various developments, and works on complex projects at EcomDev BV, where he is one of the co-founders.

---

I want to thank my girlfriend Elena for being so helpful and patient during the late nights when I was reviewing this book.

Also, I would like to appreciate the job that was done by the author of this book. Especially, after the reviews, it is always hard doing such a job. Many thanks to the team of Packt Publishing that was working on this interesting book, especially Joel Goveya, the Project Coordinator, who was very patient in spite of my short delays in sending the review.

---

**Hans Kuijpers**, an open source enthusiast was born and raised in the Netherlands and has been creating websites since 1995. He holds a BS in Technology Management from Fontys University. In the past he was working for KPN, a large Dutch Telecom company, as a developer on DNS platform.

He is currently working for Jira.nl, a Dutch company which offers Joomla! training, consulting, and project management, and Yireo.com, also located in Amsterdam, which publishes online videos and tutorials for Joomla! and Magento. He is also very active in the open source community and organizes Dutch Joomla! day events and sets up user groups. He also sets up monthly Magento user groups.

He has participated in reviewing, *Joomla! templates ontwerpen*, a Dutch book, which covers Joomla! template tutorials.

**Fernando J. Miguel** has seven years of experience in Information Technology, in which two years were dedicated as a trainee and five as a web developer. He participated in critical processes focusing on new challenges.

He has a Bachelor's degree in Information System from Centro Universitario Modulo, where he has received a scholarship for Best Academic Performance. He did his Post-Graduation in Health Informatics from the Universidade Federal de São Paulo. He has worked as an IT support volunteer for AIDS Prevention Congress in Caraguatatuba, SP, Brazil.

He has good knowledge about Content Management System tools, especially Joomla!, Magento, and WordPress. He also has knowledge about Magento e-commerce CMS development, customization, and support, and can also work on PHP development using Zend Framework.

# www.PacktPub.com

## Support files, eBooks, discount offers and more

You might want to visit [www.PacktPub.com](http://www.PacktPub.com) for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at [www.PacktPub.com](http://www.PacktPub.com) and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at [service@packtpub.com](mailto:service@packtpub.com) for more details.

At [www.PacktPub.com](http://www.PacktPub.com), you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<http://PacktLib.PacktPub.com>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

### Why Subscribe?

- ▶ Fully searchable across every book published by Packt
- ▶ Copy and paste, print and bookmark content
- ▶ On demand and accessible via web browser

### Free Access for Packt account holders

If you have an account with Packt at [www.PacktPub.com](http://www.PacktPub.com), you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.





*Dedicated to my grandfather Rafael, in his memory*



# Table of Contents

<b>Preface</b>	<b>1</b>
<b>Chapter 1: Theme Basics—Working with the Default Theme</b>	<b>9</b>
Introduction	9
Shutting down the cache before working with themes	10
Theme inheritance in Magento	11
Adding blocks	13
Creating layouts	16
Layout handles	18
Customizing templates	20
<b>Chapter 2: Working with Existing Themes</b>	<b>23</b>
Introduction	23
Installing a theme through Magento Connect	24
Installing a theme manually	26
Selecting the recently installed theme	27
Enabling template path hints	29
Simple modifications, changing a logo and header info	30
Simple modifications, link to CMS pages from our theme	32
<b>Chapter 3: Starting Our Own Theme—Basic Steps</b>	<b>35</b>
Introduction	36
Starting our theme; laying the foundation	37
Adding content in the home page	43
Adding breadcrumbs	48
Enabling the search form	50
Adding top links	53
Adding the language selector	55
Footer links	56
Adding the mini cart	58
Building our main menu	62

<b>Chapter 4: Continuing Our Theme—Other Necessary Pages</b>	<b>65</b>
Introduction	65
Enhancing what we have up to now	66
Setting the template for the catalog page	68
Setting the template for other CMS pages	74
Setting the template for the Sign In page	76
Setting the template for the Shopping Cart page	81
Modifying the product detail page	83
<b>Chapter 5: Going Further—Making Our Theme Shine</b>	<b>87</b>
Introduction	87
Using Cufón to include any font we like in our theme	88
SlideDeck content slider	90
Nivo banner slider	94
Magento Easy Lightbox	97
Adding social media sharing to product page	101
Adding featured products to the home page	105
<b>Chapter 6: Building Simple Extensions</b>	<b>113</b>
Introduction	113
Installing the ModuleCreator extension	114
Using the ModuleCreator extension	116
Building a featured products block	120
Modifying the featured products block to show the most sold products	127
<b>Chapter 7: Localization and Other Tips</b>	<b>131</b>
Introduction	131
Localizing our theme	131
Getting the current store	136
How to create a multi-currency store	140
How to create CMS pages	145
Translating the topmenu	148
<b>Chapter 8: Selling Our Theme</b>	<b>153</b>
Introduction	153
Packing our theme	153
Nice features for our theme	155
Where to sell our theme	159
Where to go from here	161

<b>Appendix: Quick Summary</b>	<b>165</b>
<b>Introduction</b>	<b>165</b>
<b>Step 1—laying the foundation</b>	<b>166</b>
<b>Step 2—small modifications that can be done in admin panel</b>	<b>168</b>
<b>Step 3—small modifications that can be done in layout files</b>	<b>172</b>
<b>Index</b>	<b>179</b>



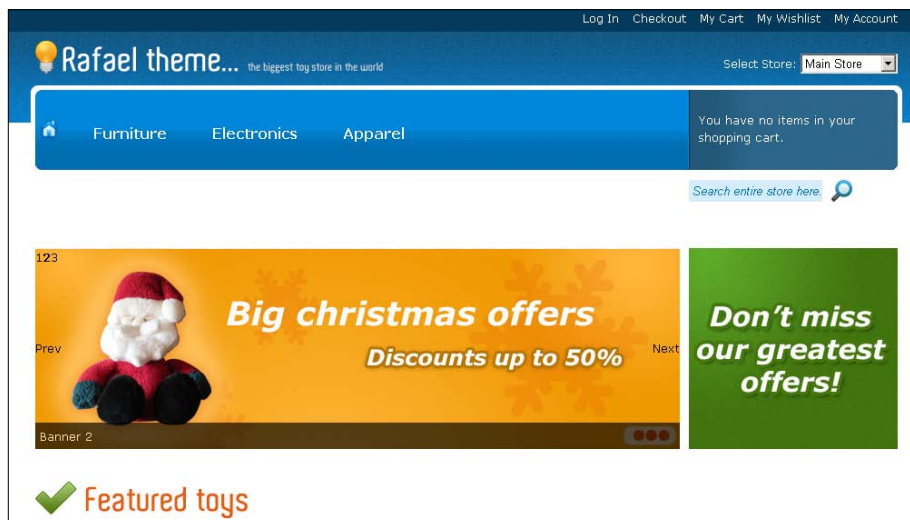
# Preface

Welcome dear reader to this Magento theming book. In this book, we are going to discover how to modify or create Magento themes from scratch. As you are reading this book, I assume you already know how great a tool this open source solution is. Though we will go through some of the basics, like installing and some theoretical concepts, we will not go into management or admin topics. And that's because we are going to centre in on how to create stunning themes for Magento.

Throughout the book we are going to work with a few different themes, the one that comes by default, one downloaded theme, and the one we are going to develop throughout the book.

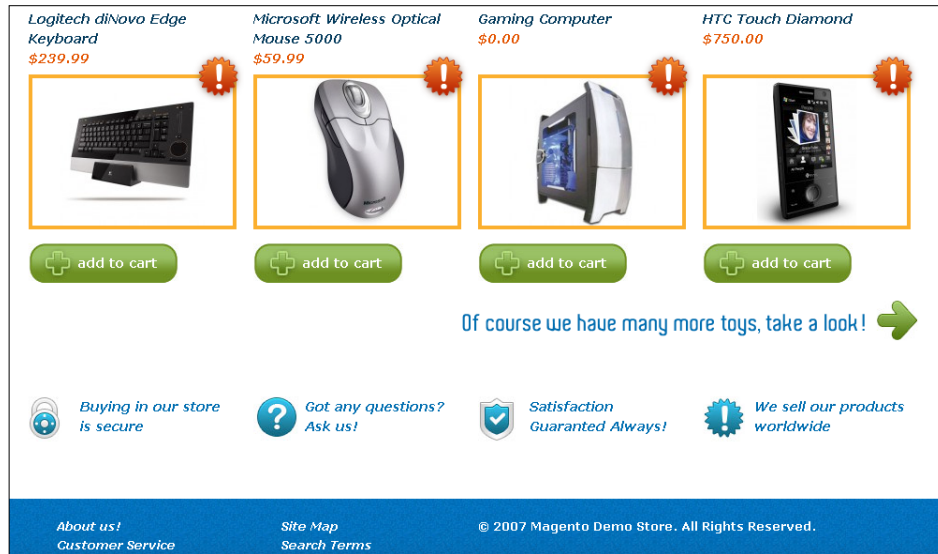
This will let us try modifications in different environments, while giving us a bit more knowledge about Magento theming. But, of course, our main task will be to create our own theme.

In many developments we could just fine installing and adapting a theme, but for others, building a theme from scratch would be required. So, why don't you take a look at the theme we are about to create through the book? Here it is:

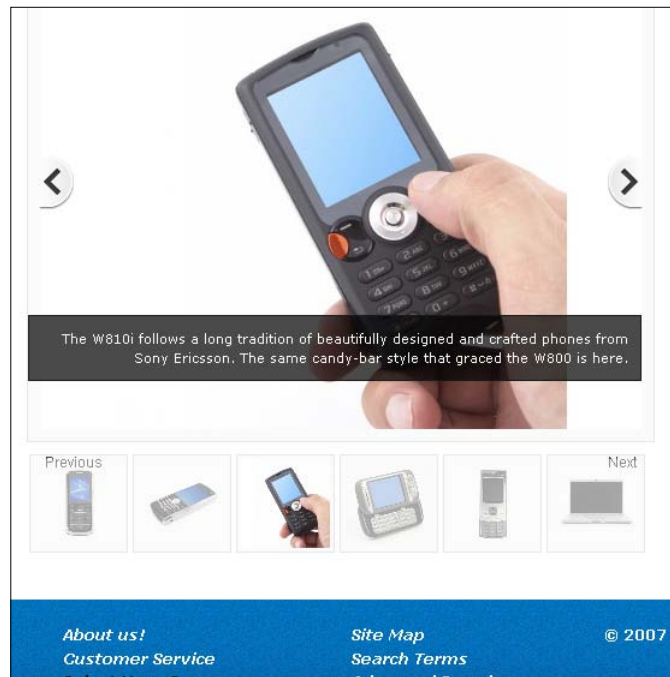




And another one:



And another one:



Great looking, isn't it? Stay with this book and you will be making themes like these in no time! You will learn how to do it in a step-by-step way separated into recipes, so you can come back later and pick the recipe that suits your needs.

Moreover, this book is structured as a Cookbook, so each recipe is fully independent from the others. You can read the chapter as a whole, going through all recipes, or, if you prefer, you can go to the recipe that interests you.

I hope you enjoy this book, at least as much as I've enjoyed writing it. I really feel the theming topic can be really fun, and also don't forget that Magento theming can be very profitable. Funny and profitable? Sounds good to me! Worrying about how much difficult this will be? Don't! We are going to go from the bottom to the top, so everyone from novice to expert can take something interesting from this book.

Also, while reading, you can take the demo theme that comes with the book and study with it. Or use it in your own projects! Want to have a quick glance at the topics we are about to see?

## What this book covers

*Chapter 1, Theme Basics—Working with the Default Theme:* This is going to be a more theoretical chapter, but don't worry, we are going to see only the concepts necessary to help you build your themes. Magento theming can be a bit tricky, but hopefully this chapter will put us on a good starting point.

*Chapter 2, Working with Existing Themes:* Sometimes we can use a preexisting theme, as a good base for our site, and modify it a bit. In this chapter, we will see how we can find a theme, install it, and make some tiny modifications to it, a good and easy way of getting a Magento installation up and running.

*Chapter 3, Starting Our Own Theme—Basic Steps:* Here starts the real purpose of the book, to create our own theme from scratch. This chapter will show us all the necessary steps, from concept to coding. I'm sure this is going to be one of your favorite chapters, so don't miss it!

*Chapter 4, Continuing Our page—Other Necessary Pages:* By this chapter we will have our theme working, but we want more. In this chapter, we will cover some interesting details that will make our theme look even better and more useful for visitors. Summarizing it, this chapter takes the theme we made in *Chapter 3* and takes it to the next level.

*Chapter 5, Going Further—Making Our Themes Shine:* In this chapter, we will be modifying other theme pages, such as the cart one, the buying process, and some others. This will help us make our theme more consistent and create a theme that really suits our needs.

*Chapter 6, Building Simple Extensions:* Though this book is not about Magento extensions, I think that covering some basics will be really helpful, as sometimes we need to include some modules in our theme. This chapter will go through the basics of extension programming. Don't miss it!

*Chapter 7, Localization and Other Tips:* Once we have our theme up and running, the next logical step is to make it available for visitors throughout the world. Localizing it is just one of the tools that will help us achieve that. This chapter will also cover other recipes that do not exactly relate to the rest.

*Chapter 8, Selling Our Theme:* Once we have our theme finished, what can we do with it? Well, selling it would be a good idea, and a profitable one! In this chapter, I will try to give you some tips about how you can make some profit out of your Magento theming skills followed by the summary of this chapter.

*Chapter 9, Appendix—Quick Summary:* This chapter is also going to be a small summary that you can follow to get started with modifying a Magento installation. Think of this chapter as a quick start guide, or a summary you can follow if you have forgotten something.

## What you need for this book

For this book we are going to use XAMPP, but any other environment should work the same; use WAMP if you prefer, or even a shared hosting. For your reference, I'm going to tell you the environment I'm going to use:

- ▶ Windows Vista.
- ▶ A copy of Magento; remember we are going to use magento-1.4.1.1, though you won't have problems using other versions.
- ▶ The sample data for Magento. This time we are going to use magento-sample-data-1.2.0.
- ▶ xampp-win32-1.7.3, as this will create a server environment, but conveniently it will be installed on our own machine.
- ▶ Some editor or IDE. You could go for IDEs such as NetBeans, but any editor would work just fine. For example, Notepad++ would be perfect for our theming purposes.
- ▶ Adobe Photoshop, GIMP, or similar.
- ▶ This book, but that's already checked!

These are the tools we are going to use; maybe at some point in time we will use something else, but these will be fine for most of our development. And most of them are free!

Of course, in any other environment you should be just fine as well, so pick the tool of your liking.

## Who this book is for

If you are a designer or programmer who wants to create excellent Magento themes, quickly and easily, this book is for you. No special knowledge of Magento or PHP is required, though some HTML and CSS experience will be helpful.

## Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "Create another empty folder named `rafael`."

A block of code is set as follows:

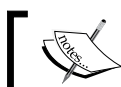
```
<default>

    <!-- Mage_Catalog -->
    <reference name="top.menu">
        <block type="catalog/navigation" name="catalog.topnav"
template="catalog/navigation/top.phtml"/>
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
    </block>
    <Paste code here>
    </reference>
```

**New terms** and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "Select **Disable** in the **Action** drop-down list."



Warnings or important notes appear in a box like this.



Tips and tricks appear like this.

## Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to [feedback@packtpub.com](mailto:feedback@packtpub.com), and mention the book title via the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the **SUGGEST A TITLE** form on [www.packtpub.com](http://www.packtpub.com) or e-mail [suggest@packtpub.com](mailto:suggest@packtpub.com).

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on [www.packtpub.com/authors](http://www.packtpub.com/authors).

## Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

## Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at <http://www.PacktPub.com>. If you purchased this book elsewhere, you can visit <http://www.PacktPub.com/support> and register to have the files e-mailed directly to you.

## Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/support>, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from <http://www.packtpub.com/support>.

## **Piracy**

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at [copyright@packtpub.com](mailto:copyright@packtpub.com) with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

## **Questions**

You can contact us at [questions@packtpub.com](mailto:questions@packtpub.com) if you are having a problem with any aspect of the book, and we will do our best to address it.



# 1

## Theme Basics— Working with the Default Theme

Ready to start Chapter 1? This chapter is going to be a bit more theoretical; don't worry, just in terms of the necessary concepts to help us while modifying or building our Magento themes. The topics we will look at are as follows:

- ▶ Shutting down the cache before working with themes
- ▶ Adding blocks
- ▶ Creating layouts
- ▶ Layout handles
- ▶ Customizing templates
- ▶ Theme inheritance in Magento

I strongly suggest that you read all of them, so we are able to create a strong foundation for the chapters to come. Magento theming can be a bit tricky at first, as some of these concepts are a bit hard to understand. With this I don't want to scare you, but point out the importance of this chapter. Ready? Let's go.

### Introduction

If you have worked on theming with other CMS or e-commerce solutions, be they open source or not, you are sure to find the process a bit different from Magento theming. For example, think about Joomla!, a very common CMS these days.



Most Joomla! templates or themes, well, at least the basic ones, only need one file to work, not taking into account the CSS, JavaScript, and so on, the so called assets. But the template itself is formed by one single file. Quite easy to work with, isn't it? We can control the appearance of all the pages in our site with one single file that is named `index.php`.

Other solutions have one file for each page or zone of our site, which is a bit harder, but still quite easy to work with. For example, we could have `home.php`, `contact.php`, `left_column.php`, and so on. When we need to create a theme for these solutions we need to edit each one of these files.

When working with Magento themes we will have three kinds of files to work with:

- ▶ Layouts
- ▶ Blocks
- ▶ Templates

We will need to work with all these kinds of files to create our theme. What is each one of them for? Well, we are about to see just that in this chapter.



The next recipe, *Shutting down the cache before working with themes*, is quite important; please read it before all other recipes in this chapter.

## Shutting down the cache before working with themes

This is a very important step, as Magento extensively uses cache in order to run more smoothly. But if we leave cache on we won't be able to see the modifications we are doing.

As we want to see what we are doing, we will shut down Magento's cache while in development.

### Getting ready

In order to do this, we need to go to our site's admin panel. Just go to the following and log in:

`http://127.0.0.1/magento/admin`



#### Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at <http://www.PacktPub.com>. If you purchased this book elsewhere, you can visit <http://www.PacktPub.com/support> and register to have the files e-mailed directly to you.

## How to do it...

Once we are logged into our site, we are able to make the necessary changes. Just follow these steps:

1. Go to the **System** menu and select the **CacheManagement** option.
2. You will see a window similar to the following screenshot:

Cache Type	Description	Associated Tags	Status
<input type="checkbox"/> Configuration	System(config.xml, local.xml) and modules configuration files(config.xml).	CONFIG	ENABLED
<input type="checkbox"/> Layouts	Layout building instructions.	LAYOUT_GENERAL_CACHE_TAG	ENABLED
<input type="checkbox"/> Blocks HTML output	Page blocks HTML.	BLOCK_HTML	ENABLED
<input type="checkbox"/> Translations	Translation files.	TRANSLATE	ENABLED
<input type="checkbox"/> Collections Data	Collection data files.	COLLECTION_DATA	ENABLED
<input type="checkbox"/> EAV types and attributes	Entity types declaration cache.	EAV	ENABLED
<input type="checkbox"/> Web Services Configuration	Web Services definition files (api.xml).	CONFIG_API	ENABLED

If we look to the extreme right, there is a column called **Status**. We will see that all cache elements are enabled. We are going to follow these steps to shut down all of them:

1. Click on the **Select All** link that appears at the top-left corner.
2. Select **Disable** in the **Action** drop-down list.
3. Click on the **Submit** button, and, just after that, **Flush Magento Cache**.

We are done; now all elements of the **Status** column will appear red, instead of green, and we will be able to see them labeled as **Disabled**.

## How it works...

We have shut down the cache; now we are able to make modifications to our themes, and see these changes without an issue.

## Theme inheritance in Magento

At this point you may be thinking that with all those files, layouts, templates, and so on, in order to build a Magento theme we will need to create a lot of files. Don't worry, that's not the case.

With the help of Magento's theme inheritance, we will only need to create the files we need to modify.

## Getting ready

This time, in order to try the inheritance capabilities of Magento, we are going to create an empty theme (fully empty), and we will see how it works without any problems.

## How to do it...

We are going to follow these steps in order to create, and select the theme:

1. Go to `app/design/frontend/default`.
2. There, create an empty folder named `rafael`, or whatever you like.
3. Go to `skin/frontend/default`.
4. Create another empty folder named `rafael`.

We are done. Now we need to select this new theme, and this can be achieved through our admin screen. So log into your Magento admin screen, and follow these instructions:

1. Go to **System** menu and select **Configuration** menu.
2. Then on the **General** tab select the **Design** option.
3. You will be able to see a screen like the following screenshot:

The screenshot shows the 'Configuration' page in the Magento admin interface, specifically the 'Design' section. It is divided into two main parts: 'Package' and 'Themes'.

**Package Section:**

- Current Package Name:** A text input field containing the value 'default'.
- + Add Exception:** An orange button with a plus icon.
- Match expressions:** A small triangle icon followed by the text 'Match expressions in the same order as displayed in the configuration.'

**Themes Section:**

This section contains a list of theme components, each with a text input field and an '+ Add Exception' button:

- Translations:** Input field is empty.
- Templates:** Input field is empty.
- Skin (Images / CSS):** Input field is empty.
- Layout:** Input field is empty.
- Default:** Input field contains the value 'rafael'.

4. In the preceding screen we can leave **default** as our current package in **Current Package Name**, but we need to change the value in Themes' **Default**. There we need to place the name of our theme, **rafael**, in our example.
5. Now click on the **Save config** button.
6. Go to the frontend of your site and refresh.

## How it works...

By this time we have created a new theme, without any single file, but our site continues to work without problem—how is that? Well, first, when we browse our site, and we reach a page, Magento will check which theme we have set as the current theme of our site. In our example it's the **Rafael** theme. Imagine we have reached a catalog page; Magento will search for the `catalog.xml` layout inside the `layout` folder of our theme. If Magento doesn't find the layout there, it will look for `catalog.xml` inside the `default` folder of the `current` interface.

If it's not found there, it will search in the `default` folder, inside the `default` interface.

If the file is not found there too it will search in the `base` interface, and `default` folder. And it will be there, as all necessary files are there.

## There's more...

Sometimes working only with CSS we will be able to create a new theme, without needing to copy any single file into our layout or templates folder.

## See also

- ▶ *Adding blocks*
- ▶ *Creating layouts*
- ▶ *Layout handles*
- ▶ *Customizing templates*

## Adding blocks

Blocks are... just that, blocks! Each block can be of one of these two types:

- ▶ Structural block
- ▶ Content block

Structural blocks are only used for positioning; they serve as containers. We can think about them much like divs, if we were to compare them to HTML tags. On the other hand, content blocks are the ones in charge of loading the necessary content. For example, we would have a menu block, a banner block, main content block, and so on.

## Getting ready

This time we are going to add a new callout to the catalog page. In order to accomplish this we are going to add a new block.

## How to do it...

We are going to make these changes in our new Rafael theme. But for now, it's empty, so we need to prepare some files first:

1. First go to `app/design/frontend/base/default/layout`.
2. Copy `catalog.xml` file.
3. Now go to `app/design/frontend/default/rafael`.
4. Create a folder named `layout`.
5. Paste the `catalog.xml` file.

Now, with this file created, we can modify it, and see our changes. Remember that we don't want to modify the base files directly:

1. Open the `catalog.xml` file.
2. Find the following code, which is more or less at line 45:

```
<default>

    <!-- Mage_Catalog -->
    <reference name="top.menu">
        <block type="catalog/navigation" name="catalog.topnav"
template="catalog/navigation/top.phtml"/>
    </reference>
    <reference name="left">
        <block type="core/template" name="left.permanent.
callout" template="callouts/left_col.phtml">
            <action method="setImgSrc"><src>images/media/col_
left_callout.jpg</src></action>
            <action method="setImgAlt" translate="alt"
module="catalog"><alt>Our customer service is available 24/7. Call
us at (555) 555-0123.</alt></action>
            <action method="setLinkUrl"><url>checkout/cart</
url></action>
        </block>
```

3. After the preceding code, add the following one:

```
<block type="core/template" name="great.savings"
template="callouts/left_col.phtml">
    <action method="setImgSrc"><src>images/media/
savings_callout.jpg</src></action>
    <action method="setImgAlt" translate="alt"
module="catalog"><alt>Great savings this winter</alt></action>
    <action method="setLinkUrl"><url>checkout/cart</
url></action>
</block>
```

4. This will create a new block, and insert a new image into it. But we need to do something more, place the image in the right place.
5. Go to `skin/frontend/default/rafael`.
6. Create a folder named `images`.
7. Create a folder inside, named `media`.
8. Place the `savings_callout.jpg` image inside the media folder.
9. Refresh the frontend; you will see something like the following image:



### How it works...

We have just created a new block in order to place a new callout. Blocks can be seen as the building parts of a Magento layout. And this recipe has shown us how easily you can add a block to our layout.

## See also

- ▶ *Creating layouts*
- ▶ *Layout handles*
- ▶ *Customizing templates*
- ▶ *Theme inheritance in Magento*

## Creating layouts

If you have read our previous recipe, *Adding blocks*, you may have learned that each Magento page is built from a number of blocks, whether they are structural or content blocks. All these blocks together build our page.

Layout files are where we place our blocks in the way we want them to appear in the page. It is here, in the layouts, that we decide which block will appear in which page. Moreover, each layout file can control the appearance of more than one page. This is one of the more interesting parts of Magento layouts, their flexibility. Let's take a look at that.

## Getting ready

In this recipe we are going to move our recently created callout, from the left column where it currently is, to the right one. And we are going to accomplish it without even touching a line of HTML!

## How to do it...

Let's go; moving these callouts is going to be easy:

1. Go to `app/design/frontend/default/rafael/layout`.
2. Open the `catalog.xml` file.
3. Search for this piece of code:

```
<reference name="left">
    <block type="core/template" name="left.permanent.
callout" template="callouts/left_col.phtml">
        <action method="setImgSrc"><src>images/media/col_
left_callout.jpg</src></action>
        <action method="setImgAlt" translate="alt"
module="catalog"><alt>Our customer service is available 24/7. Call
us at (555) 555-0123.</alt></action>
        <action method="setLinkUrl"><url>checkout/cart</
url></action>
```

```

</block>
<block type="core/template" name="great.savings"
template="callouts/left_col.phtml">
    <action method="setImgSrc"><src>images/media/
savings_callout.jpg</src></action>
    <action method="setImgAlt" translate="alt"
module="catalog"><alt>Great savings this winter</alt></action>
    <action method="setLinkUrl"><url>checkout/cart</
url></action>
</block>
</reference>

```

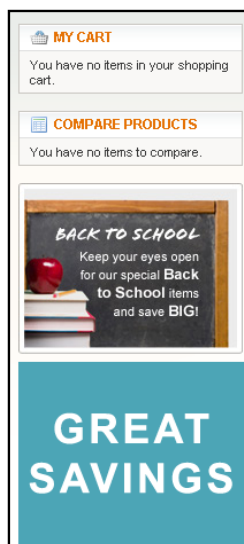
4. Cut the highlighted code, the one that refers to the savings\_callout.jpg image.
5. Paste it inside this block:

```

<reference name="right">
    <block type="catalog/product_compare_sidebar"
before="cart_sidebar" name="catalog.compare.sidebar"
template="catalog/product/compare/sidebar.phtml"/>
    <block type="core/template" name="right.permanent.
callout" template="callouts/right_col.phtml">
        <action method="setImgSrc"><src>images/media/col_
right_callout.jpg</src></action>
        <action method="setImgAlt" translate="alt"
module="catalog"><alt>Keep your eyes open for our special Back to
School items and save A LOT!</alt></action>
    </block>
    <Paste code here>
</reference>

```

6. We are done. Now we have moved the callout to the right column—check it out!





## How it works...

As we have commented before, layout files are used to create pages. They contain the blocks that will form the pages, and the structure in which way they are going to appear.

When our site needs to load the category page, our `catalog.xml` layout file will be opened, the piece of code related to the category page will be read, and the blocks inside it will indicate how the page should be built.

## See also

- ▶ *Adding blocks*
- ▶ *Layout handles*
- ▶ *Customizing templates*
- ▶ *Theme inheritance in Magento*

## Layout handles

This recipe is very much related to the layouts one. In fact, it goes a bit deeper into the explanations started in *Creating layouts* recipe. So it's a good continuation to it.

The handle is the part of the layout file which tells Magento the piece of code that needs to be loaded in order to build the page.

## Getting ready

In the previous recipe, *Creating layouts*, we moved our newly created callout from the left column to the right column. But, as it is now, the callout appears in every page of the site that has a right column. What if we want it only to appear in the catalog page?

## How to do it...

This is going to be pretty easy, so let's start:

1. Go to `app/design/frontend/default/rafael/layout`.
2. Open the `catalog.xml` file.
3. Search for the next piece of code:

```
<block type="core/template" name="great.savings"
template="callouts/left_col.phtml">
    <action method="setImgSrc"><src>images/media/
savings_callout.jpg</src></action>
```

```

        <action method="setImgAlt" translate="alt"
module="catalog"><alt>Great savings this winter</alt></action>
        <action method="setLinkUrl"><url>checkout/cart</
url></action>
    </block>

```

4. Cut the preceding piece of code.
5. Now we are going to look for the next label:  
`<catalog_category_default translate="label">`
6. At the end of that block paste the code we have cut before:

```

    <reference name="right">
        <block type="core/template" name="great.savings"
template="callouts/left_col.phtml">
            <action method="setImgSrc"><src>images/media/
savings_callout.jpg</src></action>
            <action method="setImgAlt" translate="alt"
module="catalog"><alt>Great savings this winter</alt></action>
            <action method="setLinkUrl"><url>checkout/cart</
url></action>
        </block>
    </reference>

```

7. Note we have also added the reference tag, to envelope our code. It means that this code will be placed inside the right block.

## How it works...

We are done; now our callout only appears in the catalog pages. How is that? Well it's all about the tags; for example, all the code placed inside the following tags affects all pages in our site:

```
<default>
```

But the code we place inside the following tags affects only the category pages:

```
<catalog_category_default translate="label">
```

This way we can place our blocks wherever we want, easily and without having to modify our HTML.

## There's more...

Don't worry about this concept; I know it's hard to understand it the first time you see it. But you are not alone; through the book we will be practicing such concepts and putting them into action.

## See also

- ▶ *Adding blocks*
- ▶ *Creating layouts*
- ▶ *Customizing templates*
- ▶ *Theme inheritance in Magento*

## Customizing templates

Templates are where all the HTML is placed. These are the pieces that visitors to our site will be able to see. All the other parts, layouts, blocks, and so on, are only there to organize the templates and serve them to our visitors.

## Getting ready

For this recipe we are going to create a new block, and place an image inside it. Maybe it reminds you of the block recipe, but we are going to achieve it in a different way. So try it, as you will see a different way of achieving things in Magento.

## How to do it...

This recipe involves a good number of steps, so follow them closely:

1. Go to `app/design/frontend/default/rafael/layout`.
2. Open the `catalog.xml` file.
3. Search for the following piece of code:  

```
</catalog_category_default>
```
4. Just above that code paste this one:  

```
<reference name="left">
    <block type="core/template" name="great.savings"
    template="callouts/great_savings.phtml"></block>
</reference>
```
5. The preceding piece of code will create a block, of type `core/template`. This type of code has no other functionality than loading the template attached to it in the `template` parameter, in this case the `great_savings.phtml`.
6. Now we need to create that file. Go to `app/design/frontend/default/rafael`.
7. Create a folder named `template`.

8. Inside, create a folder named `callouts`.
9. Inside, create a file named `great_savings.phtml`.
10. Inside the file place the following code:

```

```
11. And we are done. With the help of the `getSkinUrl` method we get the path to the current skin, and the rest is common HTML. It's just another way of doing things, this time working a bit more in the template part.

### How it works...

Just as layouts and blocks are used to place and order the elements when rendering a page, the templates is where all the HTML is placed. And is here where we build and create our code.

### See also

- ▶ *Adding blocks*
- ▶ *Creating layouts*
- ▶ *Layout handles*
- ▶ *Theme inheritance in Magento*



# 2

## Working with Existing Themes

Welcome to Chapter 2! This chapter is going to be a lot more practical than the previous one. This time we are going to work with already existing themes. Want to check out the topics we are about to see? Let's take a look!

- ▶ Installing a theme through Magento Connect
- ▶ Installing a theme manually
- ▶ Selecting the recently installed theme
- ▶ Enabling template path hints
- ▶ Simple modifications, changing a logo, and header info
- ▶ Simple modifications, linking to CMS pages from our theme

Don't they look interesting? So what are we waiting for? Let's get started.

### Introduction

As we have just noted, this chapter is going to be more practical. We are going to see some basic things that will help us modify the appearance of our site, from looking for free templates to the process of installing them and making some tiny changes.

I am sure this chapter helps us establish a good foundation for the chapters to come, though most of the things we are going to see are quite simple, so this is going to be a good introduction.

## Installing a theme through Magento Connect

We are going to see two ways in which we can install a new Magento theme. The first one is through Magento Connect. This involves a number of steps that we are going to follow along this recipe.

### Getting ready

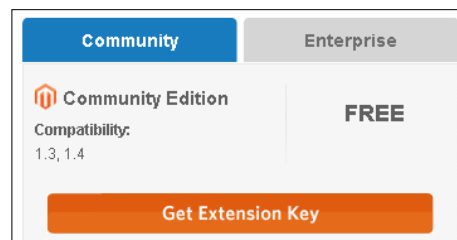
First we need to go to Magento Connect: <http://www.magentocommerce.com/magento-connect>.

There, search for the theme **Magento Classic Theme Free**, or, alternatively, load this link: <http://www.magentocommerce.com/magento-connect/TemplatesMaster/extension/928/magento-classic-theme>.

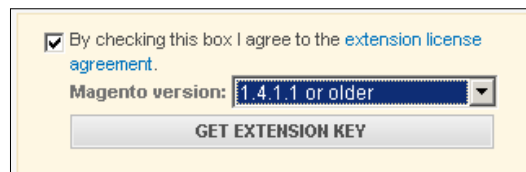
### How to do it...

This theme, the Magento Classic Theme, is the one we are going to use through this chapter. Follow these steps to install it:

1. First we need to log into our [www.magentocommerce.com](http://www.magentocommerce.com) site account. This is a necessary step to get the Extension Key.
2. The extension key can be found in a box that looks like the following screenshot:



3. We also find some other useful info there, like the compatibility version, and the price, free in this case. There's also a button that says **Get Extension Key**. Click on that button:



4. After clicking on the **Get Extension Key** button, the box that we can see in the preceding screenshot appears. On that image we must accept the license agreement, select the Magento version we are using, and again click on **Get Extension Key**:

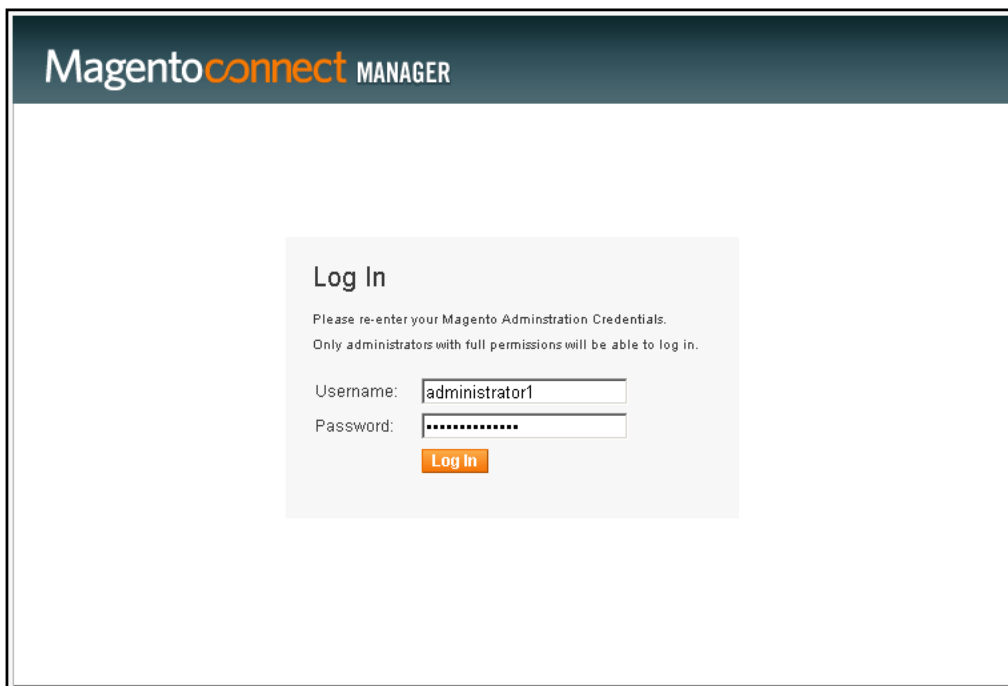


A yellow dialog box titled "What is this?" with a close button in the top right corner. It contains the text "Magento version: 1.4.1.1 or older (change)". Below this is a section titled "Extension Key" with a text input field containing "magento-community/magento\_classic\_theme\_free". At the bottom is a "SELECT" button.

5. This is the last screen. We need to obtain the extension key and copy it as we are going to use it very soon.

For the next steps we need to go to our Magento installation Admin area screen. There, carry out the following instructions:

1. Go to **System** menu, then **Magento Connect** and finally **Magento Connect Manager**.
2. The **Magento Connect Manager** screen will appear, just like the following screenshot:



The screenshot shows the "Magentoconnect MANAGER" interface. It has a dark header with the logo. The main content area is white and contains a "Log In" section. The "Log In" section has a light gray background and contains the following text: "Please re-enter your Magento Administration Credentials. Only administrators with full permissions will be able to log in." Below this text are two input fields: "Username:" with the value "administrator1" and "Password:" with a masked password "\*\*\*\*\*". At the bottom of the "Log In" section is an orange "Log In" button.



3. Here we need to insert the same login info as we used in our Magento Admin screen. In the next screen we need to insert the key we have just copied:



The screenshot shows the 'Install New Extensions' interface. It has a header bar with the title. Below it, there are two numbered steps. Step 1 says 'Search for modules via [Magento Connect](#).' Step 2 says 'Paste extension key to install:' followed by a text input field containing 'magento-community/magento\_classic\_theme\_free' and an orange 'Install' button to its right.

4. When we click on the install button the installation process starts. It will only take a few seconds and the theme will be installed into our site. Fast and easy!

## How it works...

We have just selected a theme from the Magento site, and installed it into our own Magento installation through the Magento Connect Manager. We can do this as many times as we want, and install as many themes as we need to.

## See also

- ▶ *Installing a theme manually*
- ▶ *Selecting the recently installed theme*

## Installing a theme manually

In this recipe we are going to see another method of installing themes into our Magento installation. This time we are going to see how to install a theme manually. Why is this important? Just because we may want to use themes that are not available through Magento Connect, as they can be themes purchased in other places, or downloaded from free themes.

## Getting ready

In order to follow this method we need a downloaded theme. You can get the Magento Classic theme from this link:

<http://blog.templates-master.com/free-magento-classic-theme/>

Once done, we will get a file called `f002.classic_1.zip`; just unzip it. Inside we will find many folders:

- ▶ `graphic source`: This folder has all the PSD files necessary to help us modify the theme
- ▶ `installation`: This folder is where we can find installation instructions

- ▶ `template source 1.3.2`: The theme is for Magento version 1.3.2
- ▶ `template source 1.4.0.1`: This theme is for Magento 1.4.0.1
- ▶ `template source 1.4.1.0`: This theme is for Magento 1.4.1.0

Though our Magento version is not 1.4.1.0, we could use the theme without problems. What can we find inside that folder? Another two folders:

- ▶ `app`
- ▶ `skin`

This is the folder structure we should expect from a Magento theme.

### How to do it...

Installing a theme is as easy as copying those two folders into our Magento installation, with all their sub folders and files. In our Magento installation you will see another `app` and `skin` folder, so the contents in the folder of the downloaded theme will combine with the already existing ones. No overwriting should occur. And that's all; we don't need to do anything else.

### See also

- ▶ *Installing a theme through Magento Connect*
- ▶ *Selecting the recently installed theme*

## Selecting the recently installed theme

Good, now we have installed one theme, one way or the other. So, how do we enable this theme? In fact it's quite easy; we can achieve just that from our site admin panel.

### Getting ready

If you haven't logged into your Magento installation, do it now.

### How to do it...

Well, once we are inside the admin panel of our site, we need to carry out the following steps to select our new theme:

1. Go to **System** menu, and then click on **Configuration**.
2. Look at the **General** block and select **Design**.

3. There you will see the following screenshot:

The screenshot shows a configuration window with two main sections: 'Package' and 'Themes'.  
In the 'Package' section, there is a text field labeled 'Current Package Name' containing the value 'default'. Below it is an orange button with a plus icon and the text 'Add Exception'. A small triangle icon and text below the button state: 'Match expressions in the same order as displayed in the configuration.'  
The 'Themes' section contains a list of theme categories, each with a text field and an 'Add Exception' button:

- 'Translations' with an empty text field.
- 'Templates' with an empty text field.
- 'Skin (Images / CSS)' with an empty text field.
- 'Layout' with an empty text field.
- 'Default' with the text 'f002' in the text field.

Each category has an orange 'Add Exception' button with the same match expression note as the one in the Package section.

4. There, in **Current Package Name**, we can leave it as default. In the **Themes section**, in the **Default** column, we can enter the name of the theme, **f002** for this example.
5. We are done. Click on the **Save config** button and refresh the frontend. You will be able to see the changes.

## How it works...

We are done. We have just selected the new theme. As promised it was quite easy, wasn't it? Try it yourself!

## See also

- *Installing a theme manually*
- *Installing a theme through Magento Connect*

## Enabling template path hints

Sometimes it is a bit hard to follow which block element we want to edit. For example, if we want to modify a banner, or just remove one of these blocks. Happily, Magento offers us a very useful tool we can use.

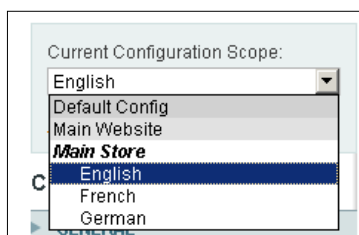
### Getting ready

Before we start, remember to log into your Magento installation admin area.

### How to do it...

Enabling these block tips is quite easy, just follow these steps:

1. Go to menu **System** and then to **Configuration**.
2. Here, we need to select the scope of the configuration. We are going to select the English store.



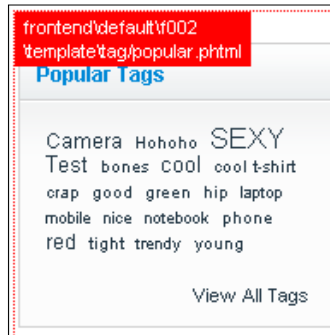
3. With this done go to the **Advanced** tab, which would be at the bottom of the page.
4. Select **Developer** and then select **Debug**.
5. Configure the options as shown in the following screenshot:

Developer Client Restrictions		
Debug		
Profiler	No	<input checked="" type="checkbox"/> Use Website [STORE VIEW]
Template Path Hints	Yes	<input type="checkbox"/> Use Website [STORE VIEW]
Add Block Names to Hints	No	<input type="checkbox"/> Use Website [STORE VIEW]

And then save the changes; that's all.

## How it works...

Now, if we reload the frontend of our site we will see red boxes giving us some information about the blocks that generated them. For example, take a look at the following screenshot:



The red box is informing us that the template file that generated this box is located at `frontend/default/f002/template/tag/popular.phtml`. We could open that file and modify it as needed.

## See also

- ▶ *Installing a theme manually*
- ▶ *Installing a theme through Magento Connect*
- ▶ *Selecting the recently installed theme*

## Simple modifications, changing a logo and header info

Once we have our site ready with its theme and so on, one of the most common tasks, or needs, is to change the logo and header info. In this recipe we are going to learn how to do that.

## How to do it...

Once in our admin screen, follow these steps:

1. Go to the **System** menu and then to **Configuration**.
2. Select **Default config** in **Current Configuration Scope**

3. Next go to the **General** tab.
4. Select the **Design** link.
5. Now, in order to change header info, go to the **HTML head** tab. It will look like the following screenshot:

HTML Head		
Default Title	<input type="text" value="Magento Commerce"/>	[STORE VIEW]
Title Prefix	<input type="text"/>	[STORE VIEW]
Title Suffix	<input type="text"/>	[STORE VIEW]
Default Description	<div>Default Description</div>	[STORE VIEW]
Default Keywords	<input type="text" value="Magento, Varien, E-commerce"/>	[STORE VIEW]

6. There are some more options that don't appear on the image, like default robots, miscellaneous scripts, and display demo store notice. Feel free to fill these values as you need.
7. The next thing we can do from this screen is to change our site logo. In order to do this, we only need to select the **Header** tab, as can be seen in the following screenshot:

Header	
Logo Image Src	<input type="text" value="images/logo.gif"/>
Logo Image Alt	<input type="text" value="Magento Commerce"/>
Welcome Text	<input type="text" value="Default welcome msg!"/>

8. Here we can indicate which image to use for the logo, as well as its **Alt** value and welcome message. Remember, the image must be placed inside our theme, that's `skin/frontend/default/f002/images`.

## How it works...

We have just learned how to modify the theme logo and header info, all done from the admin panel of our site. Those are very common changes that need to be done in almost every project.

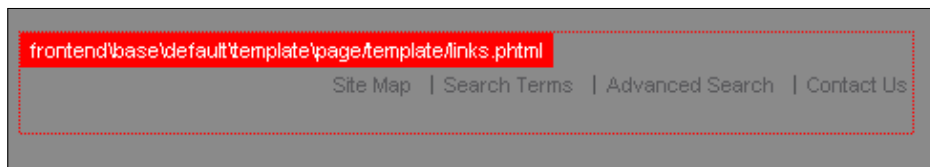
## Simple modifications, link to CMS pages from our theme

Sometimes it's necessary to add a CMS page to our site, for example an About Us page, or a FAQs page. Those things can be quite useful in an e-commerce site. In this recipe we are going to see how to link these pages from our theme.

## Getting ready

In order to follow this recipe it would be useful to enable the template path hints, and refer to *Enabling template path hints recipe* in this same chapter to learn how. For this example, we are going to add a link to the About Us page in our footer.

Take a look to the following screenshot, just to know where are we going to add the link:



## How to do it...

We can achieve this modification following these steps:

1. As the helper is telling us, we can find the PHTML file in `frontend/base/default/template/page/template/links.phtml` but note that's inside the base folder.
2. We shouldn't modify that file. It would be better to copy it and paste in `frontend/default/f002/template/page/template/links.phtml`. It may be necessary to create some folders, you can do so without facing any problem.
3. Once this is done, if we refresh our site, the tip will change, and it will indicate the path to the newly created file. Remember how inheritance worked? If you don't, check the inheritance recipe in *Chapter 1*.

4. Also open the `links.phtml` file in your favorite editor. Now we have the piece of code we need to modify, but we need the CMS page to link to.
5. Good, let's go to our admin page, there follow these steps:
  - i. Select the CMS menu.
  - ii. Click on the **Pages** option.
  - iii. There we can see a table, showing us the CMS pages that are already available in our site; we could create a new one. But we are going to use the **About Us** page.
  - iv. Look for the column **URL Key**, and copy the one for the **About Us** page. For me it's `about-magento-demo-store`, copy that. The page should look like the following screenshot:

Title	URL Key
<input type="text"/>	<input type="text"/>
About Us	about-magento-demo-store

6. Once we have the CMS page created, and the **URL Key**, we can return to our editor, where we have the `links.phtml` file loaded. There we can see this code:

```
<?php $_links = $this->getLinks(); ?>
<?php if(count($_links)>0): ?>
<ul class="links"<?php if($this->getName()): ?> id="<?php echo
$this->getName() ?>"<?php endif;?>>
    <?php foreach($_links as $_link): ?>
        <li<?php if($_link->getIsFirst() || $_link->getIsLast()):
?> class="<?php if($_link->getIsFirst()): ?>first<?php endif;
?><?php if($_link->getIsLast()): ?> last<?php endif; ?>"<?php
endif; ?> <?php echo $_link->getLiParams() ?>><?php echo $_link-
>getBeforeText() ?><a href="<?php echo $_link->getUrl() ?>"
title="<?php echo $_link->getTitle() ?>" <?php echo $_link-
>getAParams() ?>><?php echo $_link->getLabel() ?></a><?php echo
$_link->getAfterText() ?></li>
    <?php endforeach; ?>
</ul>
<?php endif; ?>
```

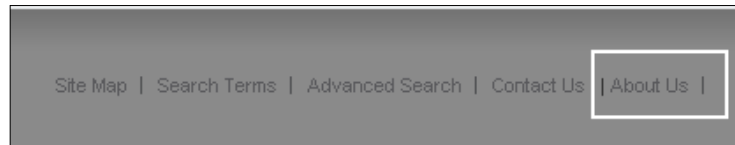
7. Just modify the preceding code to add the following highlighted one:

```
<?php $_links = $this->getLinks(); ?>
<?php if(count($_links)>0): ?>
<ul class="links"<?php if($this->getName()): ?> id="<?php echo
$this->getName() ?>"<?php endif;?>>
    <?php foreach($_links as $_link): ?>
        <li<?php if($_link->getIsFirst() || $_link->getIsLast()):
?> class="<?php if($_link->getIsFirst()): ?>first<?php endif;
?><?php if($_link->getIsLast()): ?> last<?php endif; ?>"<?php
endif; ?> <?php echo $_link->getLiParams() ?>><?php echo $_link-
>getBeforeText() ?><a href="<?php echo $_link->getUrl() ?>"
title="<?php echo $_link->getTitle() ?>" <?php echo $_link-
>getAParams() ?>><?php echo $_link->getLabel() ?></a><?php echo
$_link->getAfterText() ?></li>
    <?php endforeach; ?>
</ul>
<?php endif; ?>
```



```
<?php foreach($_links as $_link): ?>
    <li<?php if($_link->getIsFirst() || $_link->getIsLast()):
?> class="<?php if($_link->getIsFirst()): ?>first<?php endif;
?><?php if($_link->getIsLast()): ?> last<?php endif; ?>"<?php
endif; ?> <?php echo $_link->getLiParams() ?>><?php echo $_link-
>getBeforeText() ?><a href="<?php echo $_link->getUrl() ?>"
title="<?php echo $_link->getTitle() ?>" <?php echo $_link-
>getAParams() ?>><?php echo $_link->getLabel() ?></a><?php echo
$_link->getAfterText() ?></li>
    <?php endforeach; ?>
    <li>| <a href="<?php echo $this->getUrl('about-
magento-demo-store')?>">About Us</a></li>
</ul>
<?php endif; ?>
```

8. And that's all, when we refresh our page the new link will appear. Much like the following screenshot:



## How it works...

We are using the `getUrl` method, with the key we copied in order to create the link to the CMS page. Check it out! We can click on the link and the About Us page will appear.

## There's more...

CMS pages are very useful in Magento, if you haven't noticed yet; most of the content on the Home of our site comes from a CMS page. We can find it on the **CMS | Pages** menu. It appears as **Home Page**, the one which is enabled.

We can modify all its contents from here, so try this too!

# 3

## Starting Our Own Theme—Basic Steps

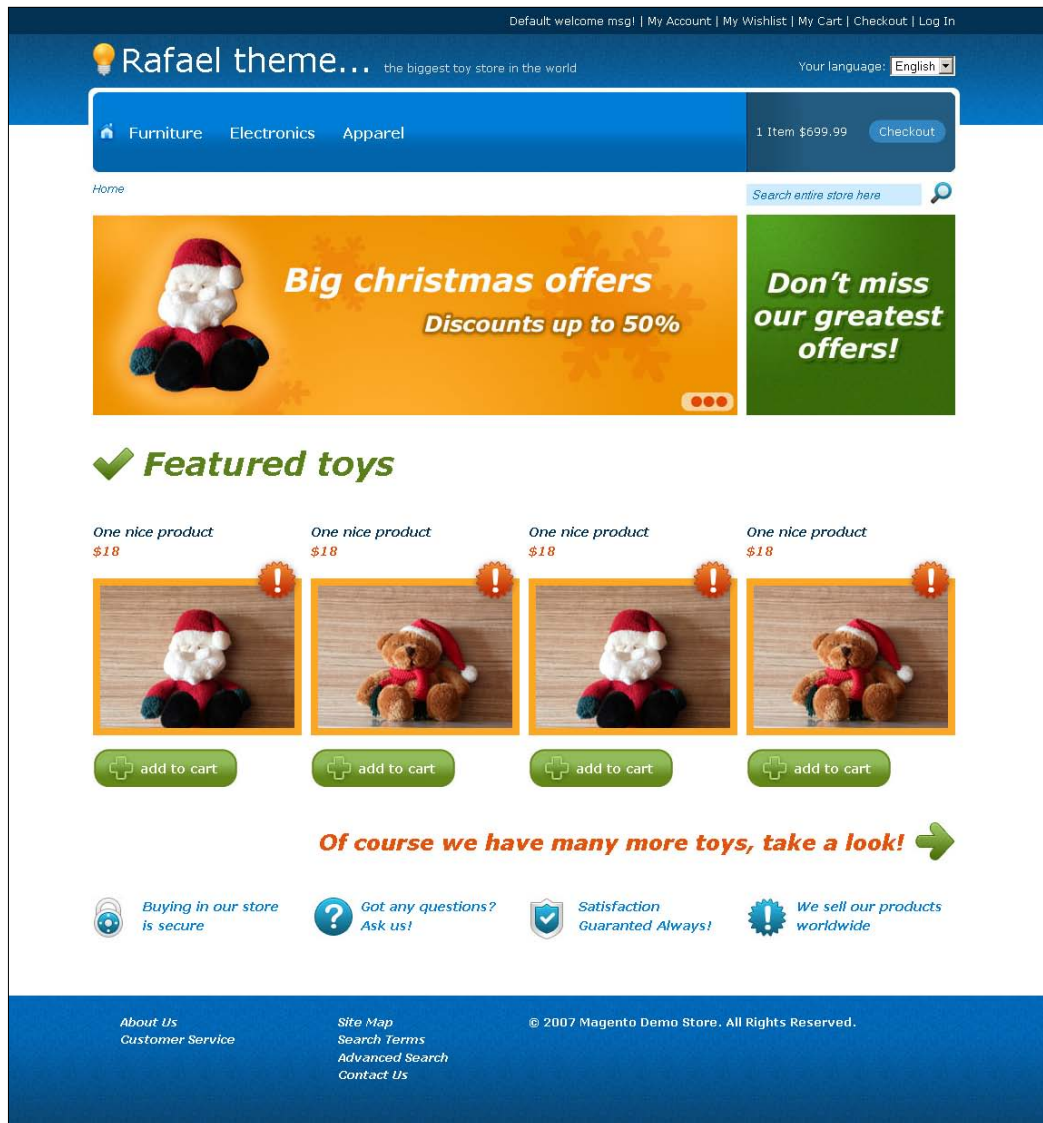
We are reaching the true purpose of this book, to build our own themes. For sure, installing and modifying existing themes is very useful. But true knowledge can only be achieved by building our own themes. Check the recipes in this chapter:

- ▶ Starting our theme; and laying the foundation
- ▶ Adding content in the home page
- ▶ Adding breadcrumbs
- ▶ Enabling the search form
- ▶ Adding top links
- ▶ Adding the language selector
- ▶ Footer links
- ▶ Adding the mini cart
- ▶ Building our main menu

With these steps I think we will cover the basic steps necessary to lay down our theme.

## Introduction

For this task we are going to need something, and that something is a theme. I've prepared a simple theme that we can use throughout the book. Take a look at its home page:



As we have commented before, this is a simple theme, but, the structure, the basic structure behind it, is not much different from other themes. So, summarizing it a bit, what we can learn here will be useful for many other themes.



The icons on this design are from <http://www.icojoy.com/articles/44/>, which has free icons. Thanks a lot to icojoy for sharing these great icons.

Get ready; we are about to start our theme!



Remember that you can find all the necessary code and files within the code bundle for the book. For example, for this chapter you can find a folder called `Home PSD`, which would contain the design, in Photoshop PSD, for the home page. Also you can find another folder called `Home HTML`, with the home page put into HTML, just as the image we have seen before. All these files are there just to make it easier for you to follow this book.

Note that though each recipe is independent to the others, sometimes I will tell you to look for a piece of code that could have been written in another recipe. Don't worry; most of the time the order of the blocks doesn't matter, and the code can be placed as you want.

Those parts in which you need to search for a previously written code is only to have some order in the code. Anyways, you can always take a look at the code bundle.

## Starting our theme; laying the foundation

In this recipe we are going to create the structure we will be using in order to create our theme. As soon as the structure is created we will be able to select our theme, though it will be limited. This way we will be able to see how our modifications affect our theme.

### Getting ready

If you would like to copy the code, instead of writing it, remember you can find all the necessary code in the code bundle in Chapter 3, recipe 1.

### How to do it...

This recipe is quite easy to follow, just follow these steps:

1. First we are going to create the folders for our theme. Go to `app/design/frontend/default` and create a folder called `rafael`.
2. Inside this folder, create three other folders:
  - ☐ `layout`
  - ☐ `locale`
  - ☐ `template`

3. Now that we have created the `theme` folder inside `app`, we need to create it inside `skin`. Go to `skin/frontend/default` and create a folder called `rafael`.
4. Inside that folder create three more:
  - ☐ `css`
  - ☐ `img`
  - ☐ `js`
5. Now that we have created the basic structure we are going to select the theme to make it active.
6. Log into your Magento admin panel.
7. Go to **System** menu, then to **Design**.
8. If there's no **Design** add a new one, and click on the **Add Design Change** button:



9. Or if a **Design Change** exists, click on it for edit:

Store	Design
<div></div>	<div></div>
Main Website Main Store English	default/v002

10. Once inside the **Design Change** screen select our newly created theme:

**General Settings**

Store \*

English

Custom Design \*

rafael

Date From

Date To

11. This is good, but if we try to refresh our site the default theme will be loaded. Why? Because of the inheritance Magento uses. Remember, we talked a bit about inheritance in *Chapter 1*. We will need to create some files first.
12. The first file we are about to create is a layout file, `page.xml`. Create it inside `app/design/frontend/default/rafael/layout`. Place the following code in it:

```
<layout version="0.1.0">
<!--
Default layout, loads most of the pages
-->

    <default translate="label" module="page">
        <label>All Pages</label>
        <block type="page/html" name="root" output="toHtml"
template="page/2columns-right.phtml">

            <block type="page/html_head" name="head" as="head">
                <action method="addJs"><script>prototype/
prototype.js</script></action>
                <action method="addJs" ifconfig="dev/js/
deprecation"><script>prototype/deprecation.js</script></action>
                <action method="addJs"><script>lib/ccard.js</script></
action>
                <action method="addJs"><script>prototype/
validation.js</script></action>
                <action method="addJs"><script>scriptaculous/
builder.js</script></action>
                <action method="addJs"><script>scriptaculous/
effects.js</script></action>
                <action method="addJs"><script>scriptaculous/
dragdrop.js</script></action>
                <action method="addJs"><script>scriptaculous/
controls.js</script></action>
                <action method="addJs"><script>scriptaculous/
slider.js</script></action>
                <action method="addJs"><script>varien/js.js</
script></action>
                <action method="addJs"><script>varien/form.js</
script></action>
                <action method="addJs"><script>varien/menu.js</
script></action>
                <action method="addJs"><script>mage/translate.js</
script></action>
                <action method="addJs"><script>mage/cookies.js</
script></action>
                <action method="addCss"><stylesheet>css/reset.
css</stylesheet></action>
                <action method="addCss"><stylesheet>css/text.css</
stylesheet></action>
```

```

        <action method="addCss"><stylesheet>css/960_24_
col.css</stylesheet></action>
        <action method="addCss"><stylesheet>css/styles.
css</stylesheet></action>
    </block>

</block>

    <block type="core/profiler" output="toHtml" name="core_
profiler"/>
    </default>

</layout>

```

13. Now we need a template file. You might have noticed that in the previous code we were making use of `2columns-right.phtml`. Now we are going to create that file `2columns-right.phtml` inside `app/design/frontend/default/rafael/template/page`. We will put the following code in it:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <?php echo $this->getChildHtml('head') ?>
</head>
<body id="web">

    <div class="container_24">

        <div id="upper_menu" class="grid_24 text_right size_12
white">
            <p>
                <a href="#">Default welcome msg!</a> | <a
href="#">My Account</a> | <a href="#">My Wishlist</a> | <a
href="#">My Cart</a> | <a href="#">Checkout</a> | <a href="#">Log
In</a>
            </p>
        </div><!-- End of the upper menu -->

        <div id="title" class="grid_24 size_12 white">
            <h1>Rafael theme... <span class="tagline">the biggest
toy store in the world</span></h1>

            <div id="language">
                Your language:

                <select>
                    <option value="#">English</option>
                </select>
            </div>
        </div><!-- End of the title -->
    </div>

```

```

<div class="clear"></div>

<div id="menu">
  <div class="grid_18">
    <ul>
      <li><a href="#">Furniture</a></li>
      <li><a href="#">Electronics</a></li>
      <li><a href="#">Apparel</a></li>
    </ul>
  </div>

  <div class="grid_6">
    <div class="mini_cart">1 Item $699.99    <span
class="checkout_link"><a href="#">Checkout</a></span></div>
  </div>
</div><!-- End of menu and mini cart -->

<div class="clear"></div>

<div id="breadcrumbs" class="grid_18 margin_bottom_10">
  <a href="#">Home</a>
</div><!-- End of breadcrumbs -->

<div id="searchbox" class="grid_6 margin_bottom_10">
  <form>
    <input type="text" name="sarch_field"
value="Search entire store here" size="27" /> <input
class="search_button" type="submit" value="Search" name="search"
/>
  </form>
</div><!-- End of search -->

<div class="clear"></div>

</div>

<div class="clear"></div><br/><br/>

<div id="footer">
  <div class="container_24">
    <div class="grid_6">
      <br/>
      <ul>
        <li><a href="#">About Us</a></li>
        <li><a href="#">Customer Service</a></li>
      </ul>
    </div>
  </div>

```



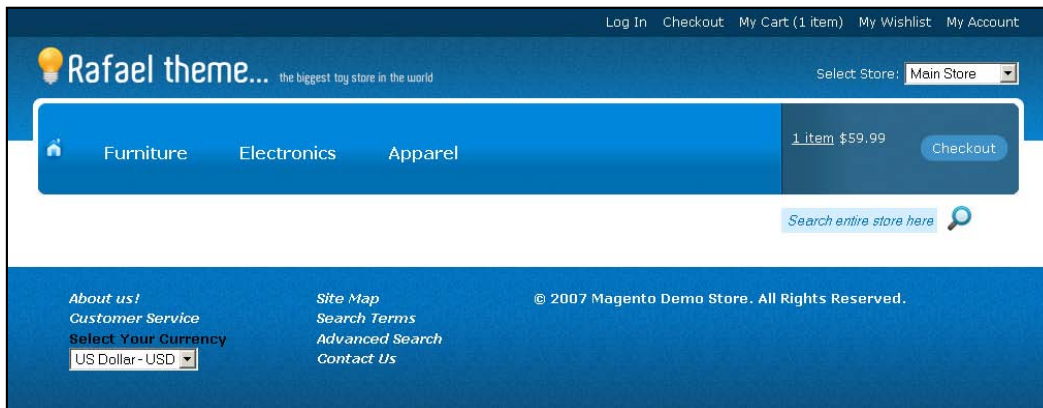
```
<div class="grid_6">
  <br/>
  <ul>
    <li><a href="#">Site Map</a></li>
    <li><a href="#">Search Terms</a></li>
    <li><a href="#">Advanced Search</a></li>
    <li><a href="#">Contact Us</a></li>
  </ul>
</div>

<!-- End of footer menus -->

<div class="grid_12">
  <br/>
  <strong class="white">© 2007 Magento Demo Store.
All Rights Reserved.</strong>
</div>
</div>
</div>

</body>
</html>
```

14. For the moment it's only HTML, without code in it. Don't worry; in time we will look into that. For now we are only placing the basic structure.
15. Now we have a layout, and a template. But we need the CSS files and the images that will help create the theme. You will find both CSS and Image files in the code bundle so copy them and paste them in the `skin/frontend/default/rafael` folder, each one in the corresponding folder.
16. Once the files are available we will be able to go to our browser and open our site. Here, we will be able to see something like the following screenshot:



## How it works...

We have just placed the basics of our theme, a layout, and a template. From now on we can work on filling the gaps and place some code into it. Which are the important things we have used in this recipe?

- ▶ The selection of the template to use:

```
<block type="page/html" name="root" output="toHtml"
template="page/2columns-right.phtml">
```

- ▶ Loading the CSS files:

```
<action method="addCss"><stylesheet>css/reset.css</stylesheet></
action>
```

- ▶ Loading header info with the `getChildHtml` method:

```
<?php echo $this->getChildHtml('head') ?>
```

## See also

- ▶ *Building our main menu*
- ▶ *Adding the mini cart*
- ▶ *Adding breadcrumbs*

## Adding content in the home page

In the previous recipe we built the envelope for our site; now we are going to add some content in the home page. By the end of this recipe our site will look almost like a real one. Get ready!

## Getting ready

You will find the necessary code for this recipe in the code bundle in Chapter 3, recipe 2.

## How to do it...

This recipe involves some coding and some work in the admin panel. Let's start:

1. First we need to open the file: `app/design/frontend/default/rafael/layout/page.xml`.
2. Here, look for this piece of code:

```
<action method="addCss"><stylesheet>css/960_24_
col.css</stylesheet></action>
```

```
<action method="addCss"><stylesheet>css/styles.
css</stylesheet></action>
</block>
```

3. Just below it add the following code:

```
<block type="core/text_list" name="content"
as="content" translate="label">
    <label>Main Content Area</label>
</block>
```

4. Now we are going to edit the template file. Open `app/design/frontend/default/rafael/template/page/2columns-right.phtml`.

5. Here, look for this:

```
</div><!-- End of search -->
```

```
<div class="clear"></div>
```

6. And after it add the following:

```
<?php echo $this->getChildHtml('content') ?>
```

7. We are done with the coding part. Now we are going to work in the admin panel of our Magento site. So log on and follow the next steps.

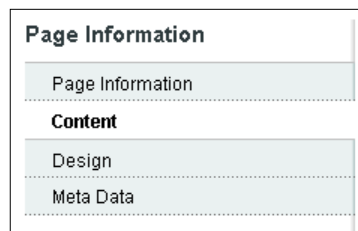
8. First go to the **CMS** menu.

9. Then to **Pages**.

10. Click on the **Home page** element (the one that looks like the following screenshot):

Home page	home	2 columns with right bar	Main Website French Store French German Store German Main Store English	Enabled
-----------	------	--------------------------	---	---------

11. Once inside, click on the **Content** tab:



12. Now we need to hide the editor, so we can paste some code. Do it by clicking the **Show/Hide Editor** button:



13. Now we can paste our code into the editor. The code we are about to paste is the following:

```
<div class="grid_18">
    <a href="#"></a>
</div><!-- Big banner -->

    <div class="grid_6">
        <a href="#"></a>
    </div><!-- Mini banner -->

<div class="clear"></div>

<div class="grid_24">
    <h1 class="tick">Featured toys</h1>
</div><!-- Big message end -->

<div class="clear"></div><br/>

<div class="grid_6">
    <p class="product_title">One nice product <br/><span
class="orange">$18</span></p>
    <div class="product_image">
        <div></div>
        <a href="#"></a>
    </div>
    <a href="#" class="price">add to cart</a>
</div>

<div class="grid_6">
    <p class="product_title">One nice product <br/><span
class="orange">$18</span></p>
    <div class="product_image">
```

```

        <div></div>
        <a href="#"></a>
    </div>
    <a href="#" class="price">add to cart</a>
</div>

<div class="grid_6">
    <p class="product_title">One nice product <br/><span
class="orange">$18</span></p>
    <div class="product_image">
        <div></div>
        <a href="#"></a>
    </div>
    <a href="#" class="price">add to cart</a>
</div>

<div class="grid_6">
    <p class="product_title">One nice product <br/><span
class="orange">$18</span></p>
    <div class="product_image">
        <div></div>
        <a href="#"></a>
    </div>
    <a href="#" class="price">add to cart</a>
</div>

<!-- End of featured products -->

<div class="clear"></div><br/><br/>

<div class="grid_24">
    <h1 class="more_products text_right"><a href="#"
class="orange_link">Of course we have many more toys, take a
look!</a></h1>
</div><!-- Big message end -->

<div class="clear"></div><br/>

```

```

<div class="grid_6">
    <h3 id="lock"><a href="#" class="blue_link">Buying in
our store is secure</a></h3>
</div>

<div class="grid_6">
    <h3 id="question"><a href="#" class="blue_link">Got
any questions? Ask us!</a></h3>
</div>

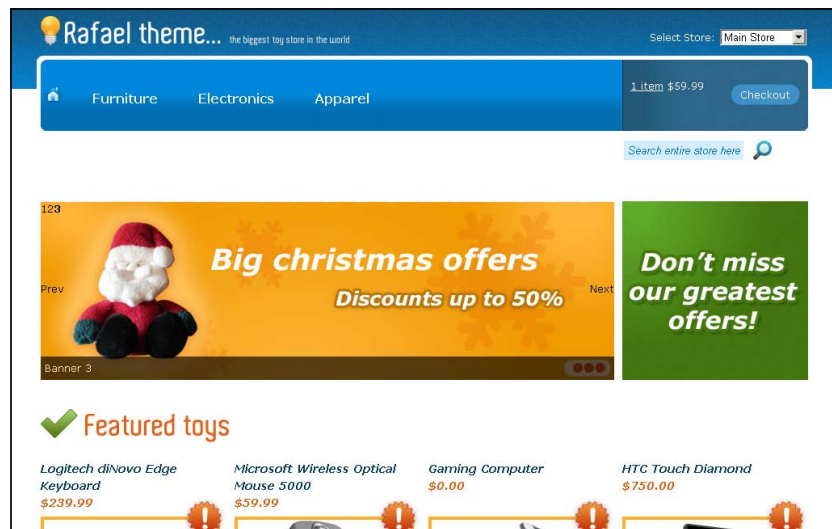
<div class="grid_6">
    <h3 id="shield"><a href="#" class="blue_
link">Satisfaction Guaranteed Always!</a></h3>
</div>

<div class="grid_6">
    <h3 id="worldwide"><a href="#" class="blue_link">We
sell our products worldwide</a></h3>
</div>

<!-- End of h3 messages -->

```

14. Now save the changes by clicking on **Save Page**. Note that this code makes use of some images; we must place them in the `skin` subfolder.
15. Create the folder `skin/frontend/default/rafael/img_products` and put the images in it. Remember, you can find these images in the code bundle, Chapter 3, recipe 2, folder `img_products`.
16. Done. We have added the code and images to have a custom home page that looks like the following screenshot:



## How it works...

In this recipe we have created a custom home page with the help of the admin panel and a bit of code. One of the most important parts of what we have just seen is the following:

1. The skin tag, so Magento can find the images:

```

```

2. The block that loads the main content in the layout:

```
<block type="core/text_list" name="content"
as="content" translate="label">
    <label>Main Content Area</label>
</block>
```

3. And the following piece of code loads the content in the template:

```
<?php echo $this->getChildHtml('content') ?>
```

## See also

- ▶ *Building our main menu*
- ▶ *Adding the mini cart*
- ▶ *Adding breadcrumbs*

## Adding breadcrumbs

Until now our page had false breadcrumbs, a static one, and pure HTML. But don't worry, we are about to solve that in this recipe.

## Getting ready

If you would like to copy the code, instead of writing it, remember you can find all the necessary files in the code bundle in Chapter 3, recipe 3.

## How to do it...

In order to add the breadcrumbs to our page, we need to modify our layout and template files. Carry out the following steps:

1. Open the file `app/design/frontend/default/rafael/layout/page.xml`.
2. Search for this piece of code:

```
<block type="core/text_list" name="content"
as="content" translate="label">
```

```

        <label>Main Content Area</label>
    </block>

```

3. Above it we are going to place the following code:

```

<block type="page/html_breadcrumbs" name="breadcrumbs"
as="breadcrumbs"/>

```

4. This will make the breadcrumbs block available, but we need to load it in the template.
5. Open the file: `app/design/frontend/default/rafael/template/page/2columns-right.phtml`.
6. There we are going to search for the div containing our false breadcrumb:

```

<div id="breadcrumbs" class="grid_18 margin_bottom_10">
    <a href="#">Home</a>
</div><!-- End of breadcrumbs ->

```

7. And replace it with a code that actually works:

```

<div id="breadcrumbs" class="grid_18 margin_bottom_10">
    <?php echo $this->getChildHtml('breadcrumbs') ?>
</div><!-- End of breadcrumbs ->

```

8. At this point our breadcrumbs would work, but the styling wouldn't be so good. We can make this better by editing our CSS files.
9. Open `skin/frontend/default/rafael/css/styles.css`.
10. Once opened look for this comment in the file:

```

/*****
*/
/***** Link styles
*****/
/*****
*/

```

11. Just above it we are going to add some styles:

```

#breadcrumbs ul{
    list-style: none;
    margin: 0;
}

#breadcrumbs ul li{
    float: left;
    margin-left: 10px;
}

```



12. We can try our breadcrumbs by going to:  
`http://127.0.0.1/magento/furniture/living-room.html`
13. They should look more or less like this:



### How it works...

Adding the breadcrumbs is quite an easy process as you only need to add the necessary block in the layout, and load it in the template.

### See also

- ▶ *Building our main menu*
- ▶ *Adding the mini cart*
- ▶ *Enabling the search form*
- ▶ *Adding top links*

## Enabling the search form

A search form is a very useful tool for our visitors, and so it's quite important for us to enable it. And that's exactly what we are going to do in this recipe.

### Getting ready

If you would like to copy the code, instead of writing it, remember you can find all the necessary code in the code bundle in Chapter 3, recipe 4.

### How to do it...

Again this recipe requires some coding, and as with previous ones it will be on our layout and template files. Let's get started:

1. Let's edit our layout file `app/design/frontend/default/rafael/layout/page.xml`.

2. Look for this piece of code:

```
<block type="page/html_breadcrumbs" name="breadcrumbs"
as="breadcrumbs"/>
```

3. And just below it we are adding the next bit:

```
<block type="core/template" name="top.search" as="topSearch"
template="catalogsearch/form.mini.phtml"/>
```

4. Now to the template part. Open the file: `app/design/frontend/default/rafael/template/page/2columns-right.phtml`.
5. Now search for the following code:

```
<form>
    <input type="text" name="sarch_field"
value="Search entire store here" size="27" /> <input
class="search_button" type="submit" value="Search" name="search"
/>
</form>
```

6. Replace it with the usual `getChildHtml` method:
7. Now, if we refresh our site, the search field will appear. But it doesn't look exactly as we want. However, we can fix this.
8. Go to `app/design/frontend/base/default/template/catalogsearch`.
9. Copy the file `form.mini.phtml`.
10. Paste it in: `app/design/frontend/default/rafael/template/catalogsearch`.
11. Now open: `app/design/frontend/default/rafael/template/catalogsearch/form.mini.phtml`.
12. Remember how Magento inheritance works. So if we copy this file, or create a new one called in the same way, the file in our theme will be used instead of the one found in the base folder.
13. Replace its contents with the following:

```
<form id="search_mini_form" action="<?php echo $this->
helper('catalogsearch')->getResultUrl() ?>" method="get">
    <div class="form-search">
        <input id="search" type="text" name="<?php echo $this->
helper('catalogsearch')->getQueryParamName() ?>" value="<?php
echo $this->helper('catalogsearch')->getEscapedQueryText() ?>"
class="input-text" />
        <button type="submit" title="<?php echo $this->__
('Search') ?>" class="search_button"><span><span><?php echo $this->
__('Search') ?></span></span></button>
```

```
<div id="search_autocomplete" class="search-  
autocomplete"></div>  
<script type="text/javascript">  
  //<![CDATA[  
    var searchForm = new Varien.searchForm('search_mini_  
form', 'search', '<?php echo $this->__('Search entire store  
here...') ?>');  
    searchForm.initAutocomplete('<?php echo $this->  
helper('catalogsearch')->getSuggestUrl() ?>', 'search_  
autocomplete');  
  //]]>  
</script>  
</div>  
</form>
```

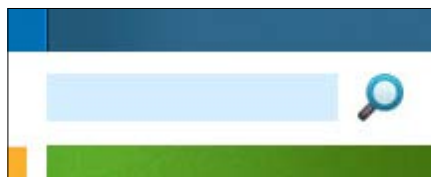
14. Now only a bit of CSS modification needs to be done.
15. Open `skin/frontend/default/rafael/css/styles.css` and search for this class declaration:

```
.search_button{  
  text-indent: -9999px;  
  width: 31px;  
  height: 27px;  
  background-color: transparent;  
  background-image: url('../img/search.jpg');  
  background-repeat: no-repeat;  
}
```

16. Add the following to it:

```
border: 0;
```

17. We are done. Our search form will look like this:



## How it works...

As in previous recipes we have first added the block to the layout and after that loaded it in the template.

## See also

- ▶ *Building our main menu*
- ▶ *Adding the mini cart*
- ▶ *Adding top links*

## Adding top links

In this recipe we are going to see how to add the top links to our theme. These links are as follows:

```
Default welcome msg! | My Account | My Wishlist | My Cart | Checkout |
Log In
```

They help our users to maintain their accounts.

## Getting ready

If you would like to copy the code, instead of writing it, remember you can find all the necessary code in the code bundle in Chapter 3, recipe 5.

## How to do it...

As in previous recipes, we need to do a little coding here, but it's very simple. Carry out the following steps:

1. Open `app/design/frontend/default/rafael/layout/page.xml`.
2. Search for the following code:
 

```
<block type="page/html_breadcrumbs" name="breadcrumbs"
as="breadcrumbs"/>
```
3. And add just before it the following one:
 

```
<block type="page/template_links" name="top.links" as="topLinks"/>
```
4. Now our work continues with the template file; open it from `app/design/frontend/default/rafael/template/page/2columns-right.phtml`.
5. Here find the top links:
 

```
<a href="#">Default welcome msg!</a> | <a href="#">My Account</a> |
<a href="#">My Wishlist</a> | <a href="#">My Cart</a> | <a
href="#">Checkout</a> | <a href="#">Log In</a>
```
6. And replace them with this code:
 

```
<?php echo $this->getChildHtml('topLinks') ?>
```

7. As before, a bit of work on CSS styles will be necessary, but only a bit. Open `skin/frontend/default/rafael/css/styles.css`.
8. Here look for the following code:

```
/*  
*/  
/****** Link styles  
******/  
/*  
*/
```

9. And above it add the following:

```
#upper_menu ul{  
    list-style: none;  
    margin-top: -20px !important;  
}  
  
#upper_menu ul li{  
    float: right;  
    margin-left: 15px;  
}
```

10. Now our top links will look like the following screenshot:



## How it works...

This recipe works much in the same way as previous recipes; first we need to load the block we want to use in the layout file:

```
<block type="page/template_links" name="top.links" as="topLinks"/>
```

And then load it in the template file:

```
<?php echo $this->getChildHtml('topLinks') ?>
```

## See also

- ▶ *Enabling the search form*
- ▶ *Adding the mini cart*
- ▶ *Building our main menu*

## Adding the language selector

This time we are going to add a language selector to our theme, helping our visitors to change their language as they need it.

### Getting ready

If you would like to copy the code, instead of writing it, remember you can find all the necessary code in the code bundle in Chapter 3, recipe 6.

### How to do it...

Quite easily in fact; the process only requires a few steps:

1. Open `app/design/frontend/default/rafael/layout/page.xml`.
2. Look for the following code:
 

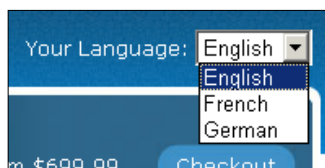
```
<block type="page/template_links" name="top.links" as="topLinks"/>
```
3. And after that add the following one:
 

```
<block type="page/switch" name="store_language" as="store_language" template="page/switch/languages.phtml"/>
```
4. Now open `app/design/frontend/default/rafael/template/page/2columns-right.phtml`.
5. Search for the following code:
 

```
Your language:

<select>
    <option value="#">English</option>
</select>
```
6. And replace it with the following one:
 

```
<?php echo $this->getChildHtml('store_language') ?>
```
7. Now we have a language selector that looks like the following screenshot:



## How it works...

Quite easily; we only need to place the block in the layout file:

```
<block type="page/switch" name="store_language" as="store_language"
template="page/switch/languages.phtml"/>
```

And next load it in the template:

```
<?php echo $this->getChildHtml('store_language') ?>
```

## See also

- ▶ *Enabling the search form*
- ▶ *Adding the mini cart*
- ▶ *Building our main menu*

## Footer links

If you have taken a look at the HTML version of our time, you may have seen some links in the footer such as site map, search terms, and so on. Of course, these links are placed only in HTML, so they aren't functional. In this recipe we are going to replace that HTML code and put in some working links.

## Getting ready

If you would like to copy the code, instead of writing it, remember you can find all the necessary in the code bundle in Chapter 3, recipe 7.

## How to do it...

In order to achieve this, follow these steps:

1. Open `app/design/frontend/default/rafael/layout/page.xml`.
2. Look for the following code:

```
</block>

<block type="core/profiler" output="toHtml" name="core_
profiler"/>
</default>
```

3. Just above it place the following code:

```
<block type="page/template_links" name="footer_links" as="footer_links" template="page/template/links.phtml"/>
```

4. The next step is the template, open it from `app/design/frontend/default/rafael/template/page/2columns-right.phtml`.
5. Search for the following code:

```
<ul>
    <li><a href="#">Site Map</a></li>
    <li><a href="#">Search Terms</a></li>
    <li><a href="#">Advanced Search</a></li>
    <li><a href="#">Contact Us</a></li>
</ul>
```

6. And replace it with the following:

```
<?php echo $this->getChildHtml('footer_links') ?>
```

7. And that's all that is necessary. Now our links look like the following screenshot:



## How it works...

Just as in previous recipes, first we need to create the block in the layout:

```
<block type="page/template_links" name="footer_links" as="footer_links" template="page/template/links.phtml"/>
```

And next load it in the template:

```
<?php echo $this->getChildHtml('footer_links') ?>
```

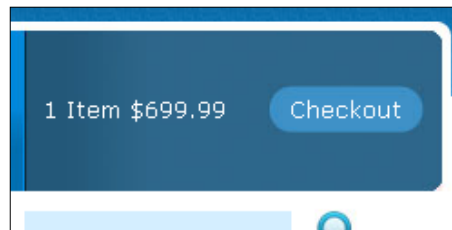
## See also

- ▶ *Adding breadcrumbs*
- ▶ *Adding top links*
- ▶ *Building our main menu*



## Adding the mini cart

It is better to have less and less parts of our theme remain pure HTML, so now it's the turn for the mini cart. This is going to require a bit more work as we want it to look like this:



The default cart that Magento uses shows more info, like product names, and so on. But for this theme we want it to show only number of items, cost, and checkout link. This way we will have the opportunity not only to place the cart, but to modify its template too.

### Getting ready

If you would like to copy the code, instead of writing it, remember you can find all the necessary in the code bundle in Chapter 3, recipe 8.

### How to do it...

This recipe requires more steps than the previous ones, but we are going to keep the steps simple. Don't worry and follow us:

1. Open `app/design/frontend/default/rafael/layout/page.xml`.
2. Look for the following code:

```
<block type="page/switch" name="store_language" as="store_language" template="page/switch/languages.phtml"/>
```
3. Just below it place the following one:

```
<block type="checkout/cart_sidebar" name="cart_sidebar" as="cart_sidebar" before="-" template="checkout/cart/sidebar.phtml"/>
```
4. Noted the `before` value? It just tells Magento that this block should appear before the one indicated in this value. In this case it's empty, so nothing happens.
5. Now to the template part. Open `app/design/frontend/default/rafael/template/page/2columns-right.phtml`.

6. Search for this code:

```
<div class="mini_cart">1 Item $699.99      <span class="checkout_
link"><a href="#">Checkout</a></span></div>
```

7. We are going to replace it with the following:

```
<div class="mini_cart"><?php echo $this->getChildHtml('cart_
sidebar') ?></div>
```

8. But this is not the end of this recipe. Still we have work to do. First we need to search for this file: `app/design/frontend/base/default/template/checkout/cart/sidebar.phtml`.

9. Copy that file and paste it in: `app/design/frontend/default/rafael/template/checkout/cart/sidebar.phtml`.

10. Once we have copied the file we are able to edit it without modifying the original. So we are going to open it and replace all its content with the following piece of code:

```
<?php if ($this->getIsNeedToDisplaySideBar()):?>
    <?php $_cartQty = $this->getSummaryCount() ?>
    <?php if ($_cartQty>0): ?>
        <?php if ($_cartQty==1): ?>
            <?php echo $this->__('<a href="%s">1 item</a>',
$this->getUrl('checkout/cart')) ?>
        <?php else: ?>
            <?php echo $this->__('<a href="%s">%s items</a>',
$this->getUrl('checkout/cart'), $_cartQty) ?>
        <?php endif ?>

        <?php echo Mage::helper('checkout')-
>formatPrice($this->getSubtotal()) ?>
        <?php if ($_subtotalInclTax = $this-
>getSubtotalInclTax()): ?>
            (<?php echo Mage::helper('checkout')-
>formatPrice($_subtotalInclTax) ?> <?php echo Mage::helper('tax')-
>getIncExcText(true) ?>)
        <?php endif; ?>

    <?php endif ?>
    <?php if($_cartQty && $this->isPossibleOnepageCheckout()): ?>
        <?php echo $this->getChildHtml('extra_actions') ?>
        <span class="checkout_link"><a href="<?php echo $this-
>getCheckoutUrl() ?>"><?php echo $this->__('Checkout') ?></a></
span>
        <?php endif ?>
        <?php $_items = $this->getRecentItems() ?>
        <?php if(!count($_items)): ?>
            <p class="empty"><?php echo $this->__('You have no items
in your shopping cart.') ?></p>
        <?php endif ?>
    <?php endif;?>
```

11. Once the code is in place, we need to work with our styles. Open `skin/frontend/default/rafael/css/styles.css`.

12. Now we are going to add the following styles:

```
.mini_cart .price{
    background: none;
    padding: 0;
}
.mini_cart a:link, .mini_cart a:visited{
    color: #ffffff;
}

.mini_cart a:hover{
    font-style:italic;
}
```

13. Search for the next piece of code:

```
.mini_cart{
    color: #ffffff;
    margin-top: 38px;
    margin-left: 10px;
}
```

14. And replace it with:

```
.mini_cart{
    color: #ffffff;
    margin-top: 28px;
    margin-left: 10px;
}
```

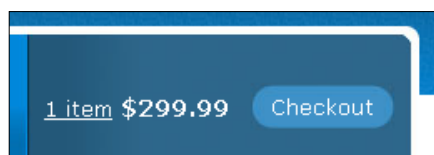
15. And lastly search for:

```
.checkout_link a:link, .checkout_link a:visited{
    background-image: url('../img/checkout_background.jpg');
    background-repeat: no-repeat;
    width: 86px;
    height: 28px;
    display: block;
    color: #ffffff;
    line-height: 28px;
    float: right;
    margin-top: -4px;
    margin-right: 10px;
    text-align: center;
    text-decoration: none;
}
```

16. And replace it with:

```
..checkout_link a:link, .checkout_link a:visited{
    background-image: url('../img/checkout_background.jpg');
    background-repeat: no-repeat;
    width: 86px;
    height: 28px;
    display: block;
    color: #ffffff;
    line-height: 28px;
    float: right;
    margin-top: 6px;
    margin-right: 10px;
    text-align: center;
    text-decoration: none;
}
```

17. Now our mini cart will look like the following screenshot:



## How it works...

Quite easily, as we have seen along the recipes of the chapter; first we need to load the block in the layout:

```
<block type="checkout/cart_sidebar" name="cart_sidebar" as="cart_
sidebar" before="-" template="checkout/cart/sidebar.phtml"/>
```

And then use it in the template:

```
<div class="mini_cart"><?php echo $this->getChildHtml('cart_sidebar')
?></div>
```

Of course after that we will need to work a bit with the mini cart template and styles, in order to make it look as we want.

## See also

- ▶ *Adding breadcrumbs*
- ▶ *Adding top links*
- ▶ *Building our main menu*
- ▶ *Enabling the search form*

## Building our main menu

This recipe is quite an important one, covering the main navigation that visitors are going to use in our site. It's going to be an easy recipe too, and will make our theme almost functional.

### Getting ready

If you would like to copy the code, instead of writing it, remember you can find all that's required in the code bundle in Chapter 3, recipe 9.

### How to do it...

This recipe involves layouts, templates, and CSS styles. But by this time, we would have practiced those tasks. Let's start:

1. First open the layout file: `app/design/frontend/default/rafael/layout/page.xml`.
2. Here, we are going to search for:  

```
<block type="checkout/cart_sidebar" name="cart_sidebar" as="cart_sidebar" before="-" template="checkout/cart/sidebar.phtml"/>
```

3. And add the following just above it:

```
<block type="core/text_list" name="top.menu" as="topMenu"/>
```

4. Good. Now to the template file. We are going to open: `app/design/frontend/default/rafael/template/page/2columns-right.phtml`.
5. In that file we need to search for our sample menu:

```
<ul>
    <li><a href="#">Furniture</a></li>
    <li><a href="#">Electronics</a></li>
    <li><a href="#">Apparel</a></li>
</ul>
```

6. And replace it with the working one:

```
<?php echo $this->getChildHtml('topMenu') ?>
```

7. But that's not the end of our recipe. We need to work on the styles too. For this recipe we are going to use the styles that come with the default template. In later chapters we will have time to explore other options, but for now that would be ok.

8. Open the file `skin/frontend/default/default/css/styles.css`. Here, copy the following code:

```

/***** < Navigation */
.nav-container { background:#0a263d url(..images/bkg_nav0.jpg)
50% 0 repeat-y; }
#nav { width:918px; margin:0 auto; padding:0 16px; font-size:13px;
}

/* All Levels */ /* Style consistent throughout all nav levels */
#nav li { position:relative; text-align:left; }
#nav li.over { z-index:998; }
#nav a,
#nav a:hover { display:block; line-height:1.3em; text-
decoration:none; }
#nav span { display:block; cursor:pointer; white-space:nowrap; }
#nav li ul span {white-space:normal; }
#nav ul li.parent a { background:url(..images/bkg_nav2.gif) 100%
100% no-repeat; }
#nav ul li.parent li a { background-image:none; }

/* 0 Level */
#nav li { float:left; }
#nav li.active a { color:#d96708; }
#nav a { float:left; padding:5px 12px 6px 8px; color:#a7c6dd;
font-weight:bold; }
#nav li.over a,
#nav a:hover { color:#d96708; }

/* 1st Level */
#nav ul li,
#nav ul li.active { float:none; margin:0; padding-bottom:1px;
background:#ecf3f6 url(..images/bkg_nav1.gif) 0 100% repeat-x; }
#nav ul li.last { background:#ecf3f6; padding-bottom:0; }

#nav ul a,
#nav ul a:hover { float:none; padding:0; background:none; }
#nav ul li a { font-weight:normal !important; }

/* 2nd Level */
#nav ul,
#nav div { position:absolute; width:15em; top:27px; left:-10000px;
border:1px solid #899ba5; }

#nav div ul { position:static; width:auto; border:none; }

/* 3rd+ Level */
#nav ul ul,
#nav ul div { top:5px; }

```

```
#nav ul li a { background:#ecf3f6; }
#nav ul li a:hover { background:#d5e4eb; }
#nav ul li a,
#nav ul li a:hover { color:#2f2f2f !important; }
#nav ul span,
#nav ul li.last li span { padding:3px 15px 4px 15px; }

/* Show menu */
#nav li ul.shown-sub,
#nav li div.shown-sub { left:0; z-index:999; }
#nav li .shown-sub ul.shown-sub,
#nav li .shown-sub li div.shown-sub { left:100px; }
/***** Navigation > */
```

9. And paste it at the end of: `skin/frontend/default/rafael/css/styles.css`.

10. We are done. Our navigation will look like the following screenshot:



## How it works...

Just as with the previous recipes, the only thing required was to add the block in the layout file:

```
<block type="core/text_list" name="top.menu" as="topMenu"/>
```

And then the templates file:

```
<?php echo $this->getChildHtml('topMenu') ?>
```

Don't forget the CSS styles.

## See also

- ▶ *Adding breadcrumbs*
- ▶ *Adding top links*
- ▶ *Adding top links*
- ▶ *Enabling the search form*

# 4

## Continuing Our Theme—Other Necessary Pages

In the previous chapter we saw some recipes that helped us in laying the foundations of our theme. We saw recipes for creating the top menu and the main menu. We also placed a minified version of the cart, and much more.

These were the basic things necessary for our site home page to work, but, what about the rest of the site? If we try to browse our site we will be able to see many pages unstyled. We are about to solve just that, using the following recipes:

- ▶ Enhancing what we have until now
- ▶ Setting the template for the catalog page
- ▶ Setting the template for other CMS pages
- ▶ Setting the template for the Sign In page
- ▶ Setting the template for the Shopping Cart page
- ▶ Modifying the product detail page

Using these recipes we will end up with a site that's almost finished, almost. Let's get started.

### Introduction

As we just read, in this chapter we are going to style many important pages, such as the catalog one, the product detail page, and many others. Some of the steps we are about to follow will be quite similar to the ones found in previous chapters, so don't worry, it's going to be easy.



## Enhancing what we have up to now

In our previous chapter we worked mainly in the home page, but as we didn't browse our site we didn't notice some faults that need to be addressed.

In this recipe we are going to work on enhancing some aspects of our previous work.

### Getting ready

If you would like to copy the code, instead of writing it, remember you can find all of the necessary files in the code bundle in Chapter 4, recipe 1.

### How to do it...

Let's get started. The sooner we end, the sooner we can continue working on other aspects of our site:

1. Copy the following file: `app/design/frontend/base/default/layout/checkout.xml`.
2. Paste it in: `app/design/frontend/default/rafael/layout/checkout.xml`.
3. Open the file and find the following piece of code:

```
<reference name="right">
    <block type="checkout/cart_sidebar" name="cart_
sidebar" template="checkout/cart/sidebar.phtml" before="-">
        <action method="addItemRender"><type>simple</
type><block>checkout/cart_item_renderer</block><template>checkout/
cart/sidebar/default.phtml</template></action>
```

4. Modify it so that it looks like this:

```
<reference name="minicart">
    <block type="checkout/cart_sidebar" name="cart_
sidebar" template="checkout/cart/sidebar.phtml" before="-">
        <action method="addItemRender"><type>simple</
type><block>checkout/cart_item_renderer</block><template>checkout/
cart/sidebar/default.phtml</template></action>
```

5. Now open: `app/design/frontend/default/rafael/layout/page.xml`. Here, look for the following piece of code:

```
<block type="core/text_list" name="content"
as="content" translate="label">
    <label>Main Content Area</label>
</block>
```

6. And, below it, add the following code:

```
<block type="core/text_list" name="right" as="right"
translate="label">
    <label>Right column</label>
</block>
```

7. Now there's a template file to edit. This one: `app/design/frontend/default/rafael/template/page/2columns-right.phtml`.
8. There, find the following piece of code:

```
<?php echo $this->getChildHtml('content') ?>
```

9. We are going to turn this code into the following:

```
<div id="content">
    <?php
        $checkhome=true;
        $page = Mage::app()->getFrontController()-
        >getRequest()->getRouteName();

        if ($page == 'cms'){
            $checkhome=(Mage::getSingleton('cms/page')-
            >getIdentifier()=='home') ? false : true;
        }

        if($checkhome){
            ?>
                <div class="grid_18">
                    <?php echo $this->getChildHtml('content') ?>
                </div>
                <div class="grid_6">
                    <?php echo $this->getChildHtml('right') ?>
                </div>
            <?php
        }else{
            echo $this->getChildHtml('content');
        }
    ?>
</div>
```

10. Good, we are done with this recipe.

## How it works...

In the previous chapter we created a `page.xml` file, and attached a template to it, but it lacked the capacity to show content in two columns.

It wasn't necessary in that chapter, as our home page had only one column. But for this one, where we want to show the content in two columns, it's going to be very necessary, and this is why we have enhanced those files.

## See also

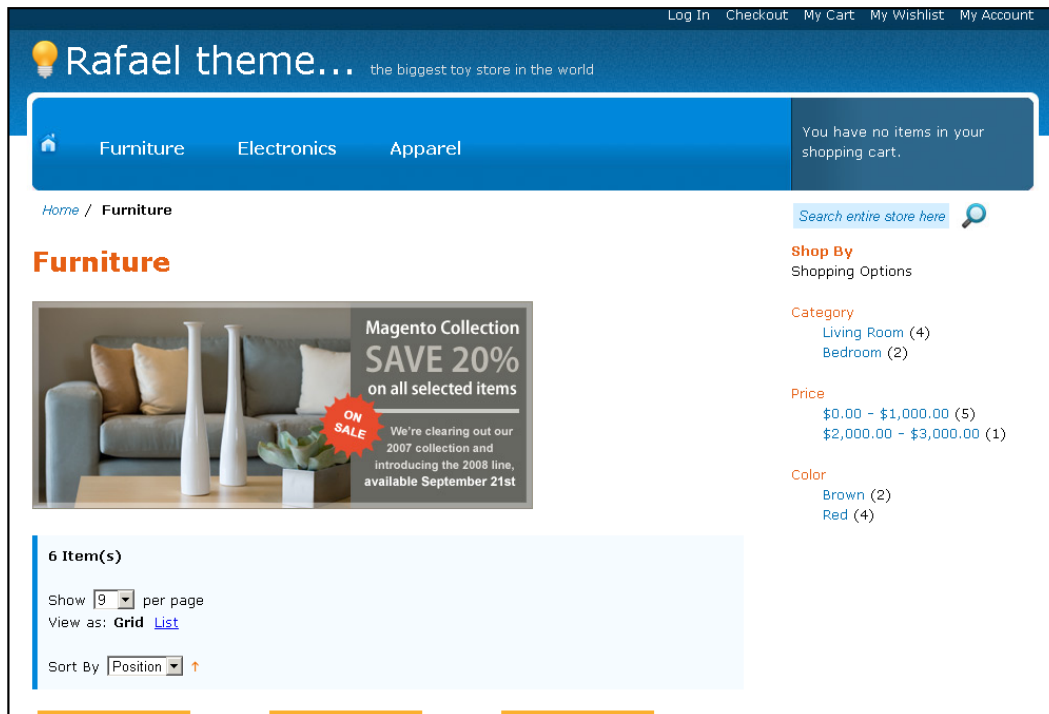
- ▶ *Setting the template for catalog page*
- ▶ *Setting the template for other CMS pages*

## Setting the template for the catalog page

The catalog page is one of the most important pages of our site, so we need to work on it. But first we are going to take a look at how it looks right now. Navigate to this URL:

`http://127.0.0.1/magento/furniture.html`

Here, we will be able to see something similar to the following screenshot:



Not so pretty, but this is not going to remain the same. By the end of this recipe this page will look much better. Let's start the real work!

## Getting ready

If you would like to copy the code, instead of writing it, remember you can find all the necessary code in the code bundle in Chapter 4, recipe 2.

## How to do it...

The steps required to style this page are as follows:

1. In order to do this, first we need to create, or duplicate, a layout file. We can take it from the base layout by copying the `app/design/frontend/base/default/layout/catalog.xml` file.
2. Paste it in: `app/design/frontend/default/rafael/layout/catalog.xml`. Now we are able to edit this file without modifying the base one, and that's what we want.
3. In that file find the following piece of code:

```
<default>

    <!-- Mage_Catalog -->
    <reference name="top.menu">
        <block type="catalog/navigation" name="catalog.topnav"
template="catalog/navigation/top.phtml"/>
    </reference>
    <reference name="left">
        <block type="core/template" name="left.permanent.
callout" template="callouts/left_col.phtml">
            <action method="setImgSrc"><src>images/media/col_
left_callout.jpg</src></action>
            <action method="setImgAlt" translate="alt"
module="catalog"><alt>Our customer service is available 24/7. Call
us at (555) 555-0123.</alt></action>
            <action method="setLinkUrl"><url>checkout/cart</
url></action>
        </block>
    </reference>
    <reference name="right">
        <block type="catalog/product_compare_sidebar"
before="cart_sidebar" name="catalog.compare.sidebar"
template="catalog/product/compare/sidebar.phtml"/>
        <block type="core/template" name="right.permanent.
callout" template="callouts/right_col.phtml">
            <action method="setImgSrc"><src>images/media/col_
right_callout.jpg</src></action>
            <action method="setImgAlt" translate="alt"
module="catalog"><alt>Keep your eyes open for our special Back to
School items and save A LOT!</alt></action>
```

```
        </block>
    </reference>
    <reference name="footer_links">
        <action method="addLink" translate="label title"
module="catalog" ifconfig="catalog/seo/site_map"><label>Site Map</
label><url helper="catalog/map/getCategoryId" /><title>Site Map</
title></action>
    </reference>
    <block type="catalog/product_price_template"
name="catalog_product_price_template" />
</default>
```

4. And replace it with the following code:

```
<default>
    <!-- Mage_Catalog -->
    <reference name="top.menu">
        <block type="catalog/navigation" name="catalog.topnav"
template="catalog/navigation/top.phtml"/>
    </reference>
    <reference name="footer_links">
        <action method="addLink" translate="label title"
module="catalog" ifconfig="catalog/seo/site_map"><label>Site Map</
label><url helper="catalog/map/getCategoryId" /><title>Site Map</
title></action>
    </reference>
    <block type="catalog/product_price_template"
name="catalog_product_price_template" />

    <reference name="right">
        <label>Right column</label>
        <remove name="paypal.partner.right.logo"/>
        <remove name="right.poll"/>
        <remove name="right.reports.product.viewed"/>
        <remove name="right.reports.product.compared"/>
        <block type="catalog/layer_view" name="catalog.
leftnav" after="currency" template="catalog/layer/view.phtml"/>
    </reference>
</default>
```

5. In the preceding code we are targeting the default page, adding to it the top links and a sitemap icon. Also targeting the `reference name="right"`, we are removing some blocks we don't want to show, such as the PayPal logo, poll, recently viewed products, and compared ones. These blocks could have been defined in any other layout file, and added to the right column. We don't need to find the layout file where we are defined. We can remove them from here. The last thing we are doing is adding the layered navigation.

6. Now if we try to reload our page:

`http://127.0.0.1/magento/furniture.html`

The right column will only show the blocks we want it to show, like the layered navigation.

7. It's time to work in the styles, just to make the page look better. Open `skin/frontend/default/rafael/css/styles.css`.
8. Add the following styles to it:

```
#content h1{
    color: #E3540E;
}

h2.product-name{
    line-height: 12px !important;
    padding: 0;
}

h2.product-name a:link, h2.product-name a:visited{
    font-size: 12px;
    text-decoration: none;
    color: #D57E13;
}

h2.product-name a:hover{
    text-decoration: underline;
}

.amount a:link, .amount a:visited, .link-wishlist:link, .link-wishlist:visited, .link-compare:link, .link-compare:visited{
    text-decoration: none;
    color: #0064AA;
}

.amount a:hover, .link-wishlist:hover, .link-compare:hover{
    text-decoration: underline;
}

ul.products-grid, ul.add-to-links{
    list-style: none;
}
```

```
li.item{
    height: 325px;
    width: 210px;
    float: left;
    margin: 20px 5px 20px 5px;
    border-bottom: 5px solid #608722;
}

li.item img{
    border: 5px solid #FBA826;
}

ul.add-to-links li{
    float: left;
    margin: 0 !important;
    padding: 0 !important;
}

#content .price-box{
    display: block;
    font-size: 18px;
    font-weight: bold;
    color: #C93D0D;
}

.btn-cart{
    background-color: #FBA826;
    border: 1px solid #C93D0D;
    color: #ffffff;
}

.toolbar{
    border-left: 5px solid #007DD7;
    padding: 10px;
    background-color: #F4FAFF;
    width: 650px;
}

.toolbar-bottom{
    clear: both;
    float: none;
}
```

9. With these styles we have covered most of the elements necessary for a common catalog page. But we need some more for the layered navigation. These are the following:

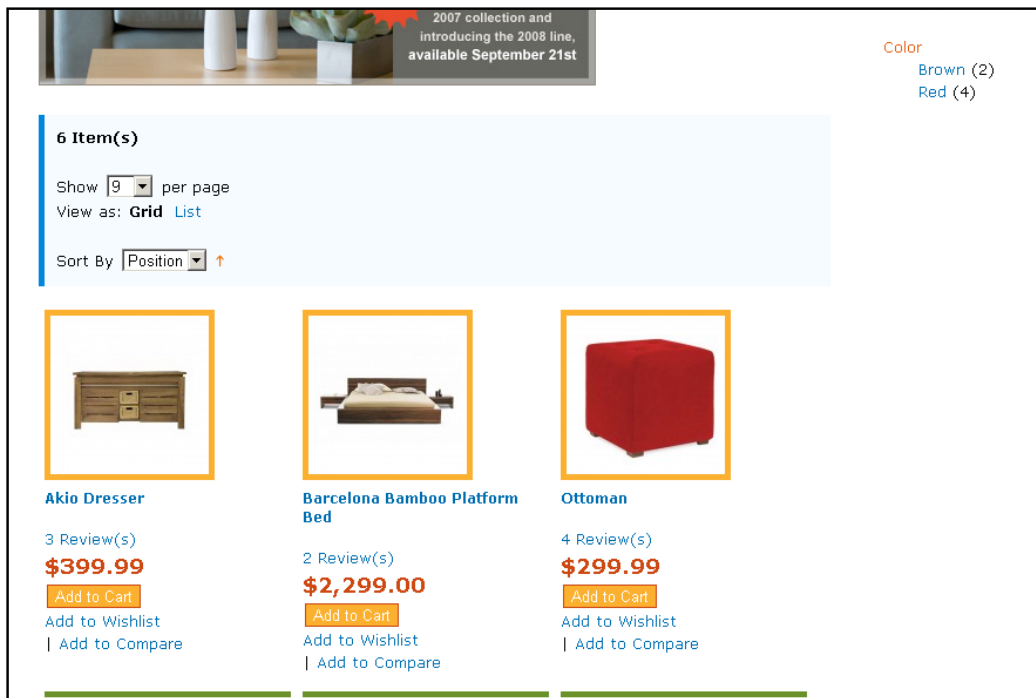
```
.block-layered-nav ol{
    list-style: none;
}

.block-layered-nav a:link, .block-layered-nav a:visited{
    color: #0064AA;
    text-decoration: none;
}

.block-layered-nav a:hover{
    text-decoration: underline;
}

.block-layered-nav .block-title, .block-layered-nav dt{
    color: #E3540E;
}
```

10. With this we have almost all the styles necessary for our catalog page covered. Our page will be looking more or less like the following screenshot:





## How it works...

In this recipe we have seen how it's possible to style the catalog pages without even modifying any layout file. However, if we want to, we are able to do so. In this case we have used the remove tag as follows:

```
<remove name="paypal.partner.right.logo"/>
```

This is done in order to remove some blocks from our right column.

## See also

- ▶ *Setting the template for other CMS pages*
- ▶ *Modifying the product detail page*

## Setting the template for other CMS pages

At this moment if we go to this URL:

```
http://127.0.0.1/magento/electronics.html
```

We will see our theme doesn't load. We will only be able to see the contents of the page, but without the other elements of our theme. What can we do for our theme to load in those pages? That's what we are about to see in this recipe.

## Getting ready

If you would like to copy the code, instead of writing it, remember you can find all the necessary code in the code bundle in Chapter 4, recipe 3.

## How to do it...

The steps required to prepare this page are as follows:

1. First we need to open our page.xml layout: app/design/frontend/default/rafael/layout/page.xml.
2. Here, we are going to add the following code:

```
<page_one_column translate="label">
  <label>All One-Column Layout Pages</label>
  <reference name="root">
    <action method="setTemplate"><template>page/2columns-
right.phtml</template></action>
```

---

```

        <!-- Mark root page block that template is applied -->
        <action method="setIsHandle"><applied>1</applied></
action>
        </reference>
    </page_one_column>

    <page_two_columns_left translate="label">
        <label>All Two-Column Layout Pages (Left Column)</label>
        <reference name="root">
            <action method="setTemplate"><template>page/2columns-
right.phtml</template></action>
            <!-- Mark root page block that template is applied -->
            <action method="setIsHandle"><applied>1</applied></
action>
            </reference>
        </page_two_columns_left>

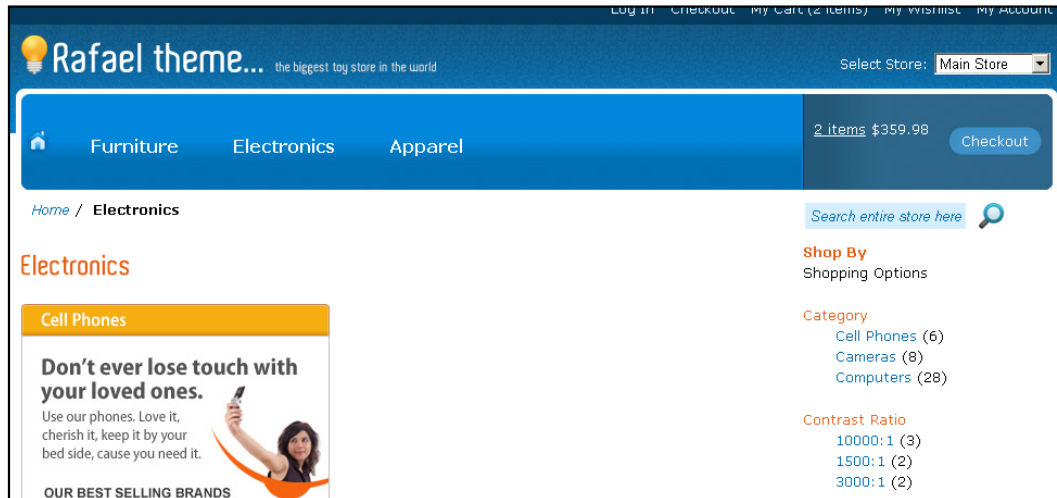
        <page_two_columns_right translate="label">
            <label>All Two-Column Layout Pages (Right Column)</label>
            <reference name="root">
                <action method="setTemplate"><template>page/2columns-
right.phtml</template></action>
                <!-- Mark root page block that template is applied -->
                <action method="setIsHandle"><applied>1</applied></
action>
                </reference>
            </page_two_columns_right>

            <page_three_columns translate="label">
                <label>All Three-Column Layout Pages</label>
                <reference name="root">
                    <action method="setTemplate"><template>page/2columns-
right.phtml</template></action>
                    <!-- Mark root page block that template is applied -->
                    <action method="setIsHandle"><applied>1</applied></
action>
                    </reference>
                </page_three_columns>

```

3. Notice that in the preceding piece of code we are targeting different page styles, and setting the template for them two columns to the right. We could have used any other template we wished, and, if we have just copied the file `page.xml` from the base folder, other pages would be called.

- This will make our CMS pages look like the following screenshot:



## How it works...

For this recipe to work we have just added some blocks, targeting different column dispositions, to our `page.xml` layout file. And, to each one of these blocks, we have set a template.

## See also

- ▶ *Enhancing what we have up to now*
- ▶ *Setting the template for the catalog page*

## Setting the template for the Sign In page

With the login, create, and account pages, the problem is that the default template is loaded instead of the one we want to use. So if we try to go to this page:

`http://127.0.0.1/magento/customer/account/create/`

We will only see an unstyled blank page. Of course, the main elements are loaded, but the page doesn't look like the rest of our site. In this recipe we are going to solve that.

## Getting ready

If you would like to copy the code, instead of writing it, remember you can find all the necessary code in the code bundle in Chapter 4, recipe 4.

## How to do it...

This recipe involves a small number of steps—ready to follow them? Let's start:

1. Our first step involves going to the following folder: `app/design/frontend/base/default/layout`.
2. Copy the `customer.xml` file.
3. And paste it in: `app/design/frontend/default/rafael/layout/customer.xml`.
4. Once we have copied the file we are going to open it in our editor.
5. Here find the following code:

```
<customer_account_login translate="label">
    ...
    <action method="setTemplate"><template>page/1column.
phtml</template></action>
    ...
</customer_account_login>
```

6. And replace it with the following:

```
<customer_account_login translate="label">
    ...
    <action method="setTemplate"><template>page/2columns-
right.phtml</template></action>
    ...
</customer_account_login>
```

7. Now we search for the next piece of code:

```
<customer_account_logoutsuccess translate="label">
    ...
    <action method="setTemplate"><template>page/1column.
phtml</template></action>
    ...
</customer_account_logoutsuccess>
```

8. And again we change it with:

```
<customer_account_logoutsuccess translate="label">
    ...
    <action method="setTemplate"><template>page/2columns-
right.phtml</template></action>
    ...
</customer_account_logoutsuccess>
```

9. There are other pieces of code that need this modification, for example:

```
<customer_account_create translate="label">
    ...
    <action method="setTemplate"><template>page/1column.
phtml</template></action>
    ...
</customer_account_create>
```

10. And we change it to a template that suits our needs, as follows:

```
<customer_account_create translate="label">
    ...
    <action method="setTemplate"><template>page/2columns-
right.phtml</template></action>
    ...
</customer_account_create>
```

11. There are many tags in this file that set templates, like the following:

```
<customer_account_forgetpassword translate="label">

<customer_account_confirmation>

<customer_account translate="label">

<customer_account_index translate="label">
```

12. In each one of these blocks we can define the template to use. This gives us a lot of flexibility.

13. We can also add some styles to this page; just open `skin/frontend/default/rafael/css/styles.css` and add the following:

```
#content ul, #content ol{
    list-style: none;
}

h2.legend{
    color: #0064AA;
}

#content a:link, #content a:visited{
    text-decoration: none;
    color: #0064AA;
}

#content a:hover{
    text-decoration: underline;
}
```

```
.input-text{
    border: 1px solid #0064AA;
}

.button{
    border: 1px solid #0064AA;
    background-color: #0064AA;
    color: #ffffff;
}
```

14. Now we are going to see an easy way to add a banner to the "create an account page".

15. First we are going to open the file `app/design/frontend/default/rafael/layout/customer.xml` and find the following piece of code:

```
<customer_account_create translate="label">
    ...
    <action method="setTemplate"><template>page/2columns-
right.phtml</template></action>
    ...
</customer_account_create>
```

16. Modify the template and set it to the following:

```
<customer_account_create translate="label">
    ...
    <action method="setTemplate"><template>page/create_
account.phtml</template></action>
    ...
</customer_account_create>
```

17. As we can see we are now setting a new template, but we have to create it, and we will do it in: `app/design/frontend/default/rafael/template/page`.

18. In that folder we are going to duplicate `2columns-right.phtml` and rename it to `create_account.phtml`.

19. Here find the following code:

```
<div id="content">
    <?php
        $checkhome=true;
        $page = Mage::app()->getFrontController()-
>getRequest()->getRouteName();

        if ($page == 'cms'){
            $checkhome=(Mage::getSingleton('cms/page')-
>getIdentifier()=='home') ? false : true;
        }
```

```

        if ($checkhome) {
            ?>
            <div class="grid_18">
                <?php echo $this->getChildHtml('content') ?>
            </div>
            <div class="grid_6">
                <?php echo $this->getChildHtml('right') ?>
            </div>
        <?php
        }else{
            echo $this->getChildHtml('content');
        }
    ?>
</div>

```

20. And change it to:

```

<div id="content">
    <div class="grid_12">
        <?php echo $this->getChildHtml('content') ?>
    </div>
    <div class="grid_12">
        <br/><br/>
        
    </div>
</div>

```

21. And we are done. Our page will look as shown in the following screenshot:

The screenshot displays the 'Rafael theme...' homepage with a navigation bar at the top containing links for 'Log In', 'Checkout', 'My Cart (2 items)', 'My Wishlist', and 'My Account'. Below the navigation bar, there are category tabs for 'Furniture', 'Electronics', and 'Apparel'. A search bar is located on the right side of the page. The main content area is titled 'Create an Account' and features a 'Personal Information' section with input fields for 'First Name', 'Last Name', and 'Email Address'. A checkbox for 'Sign Up for Newsletter' is also present. To the right of the form, there are two numbered steps: '1.) First introduce your personal info' and '2.) Now your login info'. The 'Login Information' section is partially visible at the bottom.

## How it works...

We can modify the `customer.xml` layout, and define which template each block is going to use. This gives us a lot of flexibility and an easy way to introduce different templates for each page of our site.

## See also

- ▶ *Setting the template for the Shopping Cart page*
- ▶ *Setting the template for the catalog page*

## Setting the template for the Shopping Cart page

As in other recipes of this chapter, you may find that the default template used for the checkout process is not the same that we are using. But we can solve that, as it's quite easy to establish the template to use.

## Getting ready

If you would like to copy the code, instead of writing it, remember you can find all the necessary code in the code bundle in Chapter 4, recipe 5.

## How to do it...

Get ready, we are about to define the template that our checkout process is going to use:

1. In a previous recipe we duplicated the `checkout.xml` layout, but, in case you didn't read that recipe, you can browse the `app/design/frontend/base/default/layout` folder and copy `checkout.xml` file.
2. Then paste it in: `app/design/frontend/default/rafael/layout/checkout.xml`.
3. Now we open it, and look for the following piece of code:

```
<action method="setTemplate">
```



4. We can find it in many places, like the following ones:

```
<checkout_cart_index translate="label">

<checkout_multishipping translate="label">

<checkout_onepage_index translate="label">

<checkout_onepage_success translate="label">

<checkout_onepage_failure translate="label">
```

5. In each of these places we can establish the template to use, like this:

```
<checkout_cart_index translate="label">
  <label>Shopping Cart</label>
  <remove name="right"/>
  <remove name="left"/>
  <!-- Mage_Checkout -->
  <reference name="root">
    <action method="setTemplate"><template>page/2columns-
right.phtml</template></action>
  </reference>
```

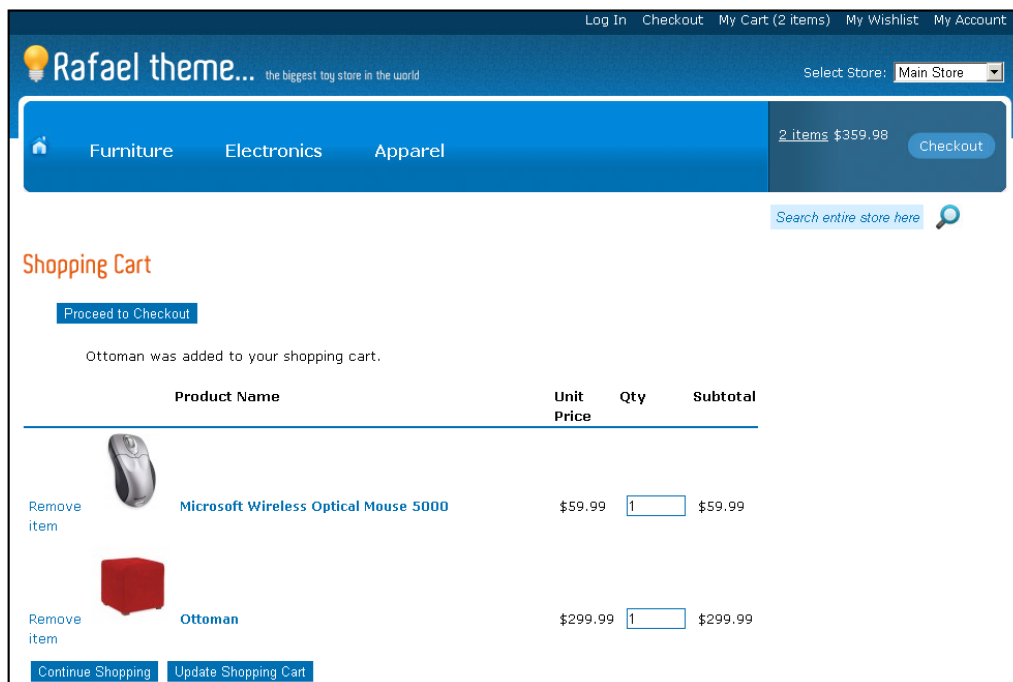
6. And we are done setting the template. It was easy, wasn't it? Well we can add some minimum styles to it; remember we place these styles in `skin/frontend/default/rafael/css/styles.css`:

```
.cart-table{
    width: 100%;
}

.cart-table td{
    padding: 5px;
}

.cart-table th{
    border-bottom: 2px solid #0065AB;
    text-align: left;
}
```

7. Now our page will look similar to the following screenshot:



## How it works...

As in previous recipes, in order to change the templates being used, we need to modify some layout files, in this case, `checkout.xml`. In these layout files we define the templates we want to use.

## See also

- ▶ *Setting the template for the catalog page*
- ▶ *Modifying the product detail page*

## Modifying the product detail page

The product detail page can easily be one of the most, if not the most, important pages of our site. Happily, by default, Magento offers us some very useful features, such as the product zoom one.

As for our site, the product zoom won't work by now, as some styles will be missing, and that's what this recipe is about, setting the zoom to work.

## Getting ready

If you would like to copy the code, instead of writing it, remember you can find all the necessary code in the code bundle in Chapter 4, recipe 6.

## How to do it...

This recipe is quite easy, just follow these steps:

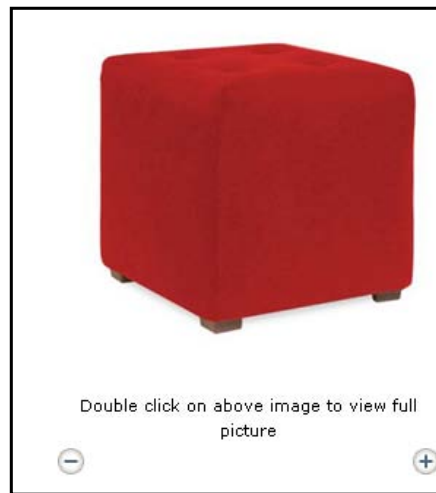
1. First we need to open our CSS file `skin/frontend/default/rafael/css/styles.css`.
2. Once we have our file opened we paste the following styles into it:

```
/* Product Images */
.product-view .product-img-box { float:left; width:267px; }
.col3-layout .product-view .product-img-box { float:none; margin:0
auto; }
.product-view .product-img-box .product-image { margin:0 0 13px; }
.product-view .product-img-box .product-image-zoom {
position:relative; width:265px; height:265px; overflow:hidden;
z-index:9; }
.product-view .product-img-box .product-image-zoom img {
position:absolute; left:0; top:0; cursor:move; }
.product-view .product-img-box .zoom-notice { font-size:11px;
margin:0 0 5px; text-align:center; }
.product-view .product-img-box .zoom { position:relative;
z-index:9; height:18px; margin:0 auto 13px; padding:0 28px;
background:url(../images/slider_bg.gif) 50% 50% no-repeat;
cursor:pointer; }
.product-view .product-img-box .zoom.disabled { -moz-opacity:.3;
-webkit-opacity:.3; -ms-filter:"progid:DXImageTransform.Microsoft.
Alpha(Opacity=30)";/*IE8*/ opacity:.3; }
.product-view .product-img-box .zoom #track { position:relative;
height:18px; }
.product-view .product-img-box .zoom #handle { position:absolute;
left:0; top:-1px; width:9px; height:22px; background:url(../
images/magnifier_handle.gif) 0 0 no-repeat; }
.product-view .product-img-box .zoom .btn-zoom-out {
position:absolute; left:2px; top:0; }
.product-view .product-img-box .zoom .btn-zoom-in {
position:absolute; right:2px; top:0; }
.product-view .product-img-box .more-views h2 { font-
size:11px; border-bottom:1px solid #ccc; margin:0 0 8px; text-
transform:uppercase; }
```

```
.product-view .product-img-box .more-views ul { margin-left:-9px }
.product-view .product-img-box .more-views li { float:left;
margin:0 0 8px 9px; }
.product-view .product-img-box .more-views li a { float:left;
width:56px; height:56px; border:2px solid #ddd; overflow:hidden; }

.product-image-popup { margin:0 auto; }
.product-image-popup .nav { font-weight:bold; text-align:center; }
.product-image-popup .image { display:block; margin:10px 0; }
.product-image-popup .image-label { font-size:13px; font-
weight:bold; margin:0 0 10px; color:#2f2f2f; }
```

3. These styles come from the default theme, and it will be made possible for the zoom image feature to work, as we can see in the following screenshot:



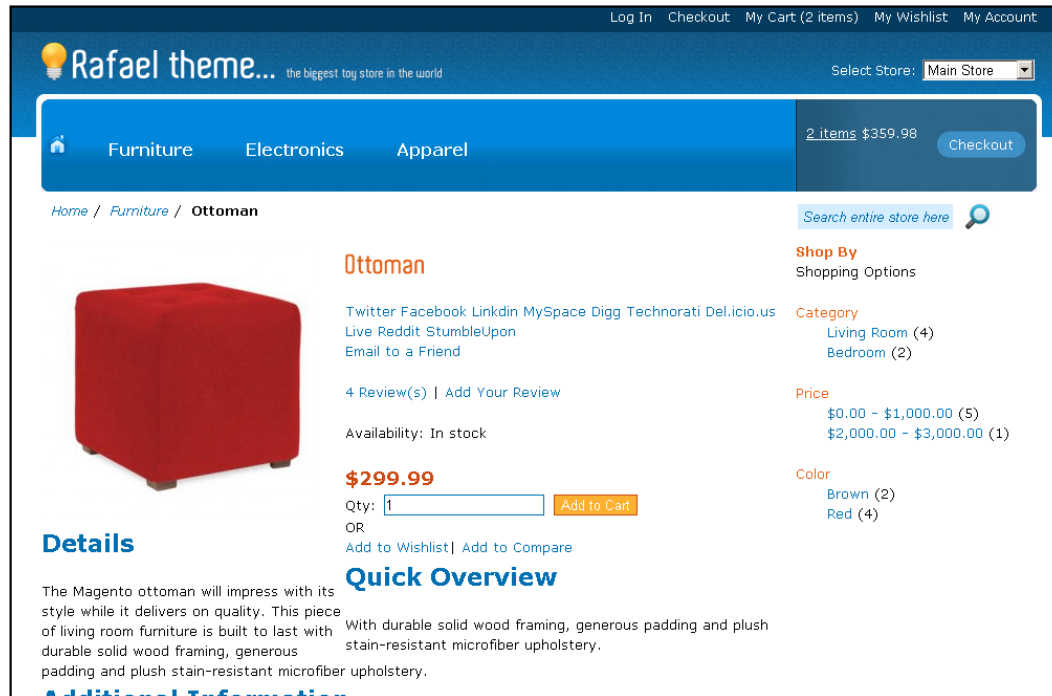
4. Also add some other styles to the CSS file:

```
.product-tags li{
    float: left;
}

#content h2{
    color: #0064AA;
}

.short-description, .product-shop{
    width: 420px;
    float: right;
}
```

5. And with that our product page will look like this:



6. And we are done for now, but don't worry, we will see more recipes in following chapters to keep enhancing this page.

## How it works...

In this recipe we have added some styles to our `styles.css` file. These styles were necessary for the product zoom feature to work properly, and we were able to get them from the default theme `styles.css` file.

## See also

- ▶ *Setting the template for catalog page*
- ▶ *Setting the template for other CMS pages*

# 5

## Going Further—Making Our Theme Shine

Welcome to *Chapter 5*! In this chapter we are going to play with some nice effects, slideshows, fonts, and much more. All these things will make our theme even more interesting and attractive for our visitors.

This is going to be a very practical chapter, as well as a very useful one, as quite often these things help our site to achieve even more sales. Let's take a look at the things we are about to cover in more detail:

- ▶ Using Cufon to include any font we like in our theme
- ▶ SlideDeck content slider
- ▶ Nivo Banner Slider
- ▶ Magento Easy Lightbox
- ▶ Adding social media sharing to product page
- ▶ Adding featured products to the home page

Looks quite interesting, doesn't it? Let's get started!

### Introduction

As in previous chapters, you will find all the necessary code in the code bundle; however, some JavaScript libraries won't be included. Don't worry, they are very easy to find and download!

## Using Cufón to include any font we like in our theme

This technique, which allows us to use any font we like in our site, is quite common these days, and is very, very useful. Years ago, if we wanted to include some uncommon font in our design, we could only do it by using a substitutive image. Nowadays, we have many options, such as some PHP extensions, @font-face, Flash (sIFR), and, of course, Cufón. And that last option is the one we are about to use.

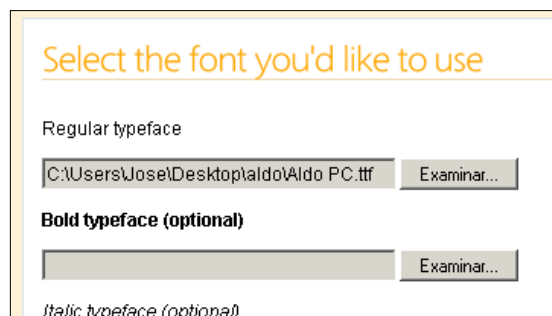
### Getting ready

If you would like to copy the code, instead of writing it, remember you can find all the necessary code in the code bundle in Chapter 5, recipe 1.

### How to do it...

Ready to start this first recipe? Let's go:

1. First we have to decide which font to use; this is totally up to you. In this example I'm going to use this one: <http://www.dafont.com/aldo.font>. It's a free font, and looks very good! Thanks to Sacha Rein. Just download it or find any other font of your liking.
2. Now we need to use the Cufón generator in order to create the font package. We will go to this URL: <http://cufon.shoqolate.com/generate/>. There we will be able to see something similar to the following screenshot:

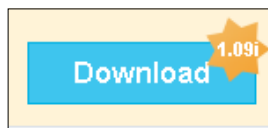


The screenshot shows a web form titled "Select the font you'd like to use". It has three sections: "Regular typeface" with a text input field containing "C:\Users\Jose\Desktop\aldo\Aldo PC.ttf" and an "Examinar..." button; "Bold typeface (optional)" with an empty text input field and an "Examinar..." button; and "Italic typeface (optional)" which is currently empty.

3. On that screen we only have to follow all the instructions and click on the **Let's do this** button:

Let's do this!

4. This will create a file called `Aldo_600.font.js`, which will be the one we will be using in this example. But we also need something more, the Cufón library. Download it by clicking the **Download** button:



5. After downloading the Cufón library, we will download a file called `cufon-yui.js`.
6. We now have all the necessary tools. It's time to make use of them.
7. First place `cufon-yui.js` and `Aldo_600.font.js` inside `skin/frontend/default/rafael/js`.
8. Now we need to edit our layout file. Let's open: `app/design/frontend/default/rafael/layout/page.xml`.
9. Once we have this file opened we are going to look for this piece of code:

```
<action method="addCss"><stylesheet>css/960_24_col.css</stylesheet></action>
<action method="addCss"><stylesheet>css/styles.css</stylesheet></action>
```

10. And just below it we are going to add the following:

```
<action method="addJs"><script>../skin/frontend/default/rafael/js/cufon-yui.js</script></action>
<action method="addJs"><script>../skin/frontend/default/rafael/js/Aldo_600.font.js</script></action>
```

See, we are adding our two JavaScript files. The path is relative to Magento's root JavaScript folder.

11. With this done, our theme will load these two JS files. Now we are going to make use of them. Just open `app/design/frontend/default/rafael/template/page/2columns-right.phtml`.
12. At the bottom of this file we are going to add the following code:

```
<script type="text/javascript">
    Cufon.replace('h1');
</script>

</body>
</html>
```



13. And that's all we need. Check the difference! Before we had something like the following:



14. And now we have this:



Every h1 tag of our site will now use our Aldo font, or any other font of your liking. The possibilities are endless!

### How it works...

This recipe was quite easy, but full of possibilities. We first downloaded the necessary JavaScript files, and then made use of them with Magento's add JS layout tag. Later we were able to use the libraries in our template as in any other HTML file.

### See also

- ▶ *Magento Easy Lightbox*
- ▶ *Adding featured products to the home page*

## SlideDeck content slider

Sometimes our home page is crowded with info, and we still need to place more and more things. A great way of doing so is using sliders. And the SlideDeck one offers a very configurable one. In this recipe we are going to see how to add it to our Magento theme. You shouldn't miss this recipe!

### Getting ready

If you would like to copy the code, instead of writing it, remember you can find all the necessary code in the code bundle in Chapter 5, recipe 2.

## How to do it...

Making use of the slider in our theme is quite simple, let's see:

1. First we need to go to <http://www.slidedeck.com/> and download SlideDeck Lite, as we will be using the Lite version in this recipe.
2. Once downloaded we will end up with a file called `slidedeck-1.2.1-lite-wordpress-1.3.6.zip`. Unzip it.
3. Go to `skin/frontend/default/rafael/css` and place the following files in it:

- ☐ `slidedeck.skin.css`
- ☐ `slidedeck.skin.ie.css`
- ☐ `back.png`
- ☐ `corner.png`
- ☐ `slides.png`
- ☐ `spines.png`

4. Next, place the following files in `skin/frontend/default/rafael/js`:

- ☐ `jquery-1.3.2.min.js`
- ☐ `slidedeck.jquery.lite.pack.js`

5. Good, now everything is in place. Next, open: `app/design/frontend/default/rafael/layout/page.xml`.
6. Now look for the following piece of code:

```
<block type="page/html_head" name="head" as="head">
    <action method="addJs"><script>prototype/
prototype.js</script></action>
    <action method="addJs" ifconfig="dev/js/
deprecation"><script>prototype/deprecation.js</script></action>
    <action method="addJs"><script>lib/ccard.js</script></
action>
```

7. Place the jQuery load tag just before the prototype one, so we have no conflict between libraries:

```
<block type="page/html_head" name="head" as="head">
    <action method="addJs"><script>../skin/frontend/
default/rafael/js/jquery-1.3.2.min.js</script></action>
    <action method="addJs"><script>prototype/
prototype.js</script></action>
    <action method="addJs" ifconfig="dev/js/
deprecation"><script>prototype/deprecation.js</script></action>
```

8. Also add the following code before the closing of the block:

```
<action method="addCss"><stylesheet>css/slidedeck.skin.css</stylesheet></action>
<action method="addCss"><stylesheet>css/slidedeck.skin.ie.css</stylesheet></action>
<action method="addJs"><script>../skin/frontend/default/rafael/js/slidedeck.jquery.lite.pack.js</script></action>
```

9. The following steps require us to log in to the administrator panel. Just go to:

<http://127.0.0.1/magento/index.php/admin>.

10. Now to CMS/Pages, open the home page one:

Home page	home	2 columns with right bar	Main Website Main Store English French German	Enabled
-----------	------	--------------------------	---	---------

11. Once inside, go to the **Content** tab.

12. Click on **Show/Hide** editor.

13. Find the following code:

```
<div class="grid_18">
  <a href="#"></a>
</div><!-- Big banner -->
```

14. And replace it with the following:

```
<div class="grid_18">
  <style type="text/css">
    #slidedeck_frame {
      width: 610px;
      height: 300px;
    }
  </style>

  <div id="slidedeck_frame" class="skin-slidedeck">
    <dl class="slidedeck">
      <dt>Slide 1</dt>
      <dd>Sample slide content</dd>
      <dt>Slide 2</dt>
      <dd>Sample slide content</dd>
    </dl>
  </div>
  <script type="text/javascript">
```

```

        jQuery(document).ready(function($) {
            $('.slidedeck').slidedeck();
        });

</script>
<!-- Help support SlideDeck! Place this noscript tag on
your page when you deploy a SlideDeck to provide a link back! -->
<noscript>
    <p>Powered By <a href="http://www.slidedeck.com"
title="Visit SlideDeck.com">SlideDeck</a></p>
</noscript>

</div>
<!-- Big banner ->

```

15. Save the page and reload the frontend.

16. Our front page should have a slider just like the one seen in the following screenshot:



17. And we are done! Remember you can place any content you want inside these tags:

```

<dl class="slidedeck">
  <dt>Slide 1</dt>
  <dd>Sample slide content</dd>
  <dt>Slide 2</dt>
  <dd>Sample slide content</dd>
</dl>

```

## How it works...

This recipe is also quite easy. We only need to load the necessary JavaScript files and edit the content of the home page, in order to add the necessary code.

## See also

- ▶ *Nivo banner slider*
- ▶ *Adding featured products to the home page*

## Nivo banner slider

Now we'll look at another slider, this time one of my favorites, the Nivo slider. This slider has many effects and options to configure it, a good feature for any website. In this recipe we are going to see how to use this powerful script.

## Getting ready

If you would like to copy the code, instead of writing it, remember you can find all the necessary code in the code bundle in Chapter 5, recipe 3.

## How to do it...

Let's get started with this recipe. Follow these steps:

1. First we need to download the script. We can find it here:  
<http://nivo.dev7studios.com/>.
2. Once downloaded we should have a file called `nivo-slider2.4.zip`. We will unzip it.
3. Inside we will find a file called `jquery.nivo.slider.pack.js`. Copy it and paste on `skin/frontend/default/rafael/js/jquery.nivo.slider.pack.js`.
4. Also, inside the ZIP file we can find a file called `nivo-slider.css`; we will copy it to `skin/frontend/default/rafael/css/nivo-slider.css`.
5. We will also need a copy of jQuery. Make sure to have a copy of it in `skin/frontend/default/rafael/js`. In this example I'm going to use `jquery-1.4.3.min.js`.
6. Once we have every file in place it's time to start working in our Magento files.
7. Now open `app/design/frontend/default/rafael/layout/page.xml`.

8. Find the following piece of code:

```
<default translate="label" module="page">
    <label>All Pages</label>
    <block type="page/html" name="root" output="toHtml"
        template="page/2columns-right.phtml">

        <block type="page/html_head" name="head" as="head">
```

9. Just below this add the following:

```
<action method="addJs"><script>../skin/frontend/default/rafael/js/
jquery-1.4.3.min.js</script></action>
```

10. And finally, before the closing of the block, we are going to add the following code:

```
        <action method="addCss"><stylesheet>css/nivo-
slider.css</stylesheet></action>
        <action method="addJs"><script>../skin/frontend/
default/rafael/js/jquery.nivo.slider.pack.js</script></action>
    </block>
```

11. Now we are ready to start making use of our script. And we are going to do that from our admin panel.

12. Log into your Magento admin panel: <http://127.0.0.1/magento/index.php/admin>

13. Now go to the **CMS** menu, then **Pages**, and select the home page:

Home page	home	2 columns with right bar	Main Website Main Store English French German	Enabled
-----------	------	--------------------------	---	---------

14. Once inside select the **Content** tab:

**Page Information**

Page Information

**Content**

Design

Meta Data

15. And then we click on the **Show/Hide Editor** button:



16. Now we are going to replace the big banner code:

```
<div class="grid_18"><a href="#"></a></div>
```

With the following:

```
<div class="grid_18">
  <div id="slider">
    
    
    
  </div>
  <script type="text/javascript">
    jQuery(document).ready(function($) {
      $('#slider').nivoSlider({
        effect:'sliceUp',
        slices:15,
        animSpeed:800,
        pauseTime:6000,
        directionNav:true, //Next & Prev
        directionNavHide:false, //Only show on hover
        controlNav:true, //1,2,3...
        pauseOnHover:false, //Stop animation while hovering
        captionOpacity:.7 //Universal caption opacity
      });
    });
  </script>
</div>
```

17. Don't forget to save the changes.

18. Now go to the front page and reload; we will be able to see something similar to the following screenshot:



## How it works...

Making the script work is quite simple; first we need to load the necessary assets with `addJs` and `addCss`:

```
<action method="addJs"><script>../skin/frontend/default/rafael/js/
jquery-1.4.3.min.js</script></action>
<action method="addCss"><stylesheet>css/nivo-slider.css</
stylesheet></action>
<action method="addJs"><script>../skin/frontend/default/rafael/js/
jquery.nivo.slider.pack.js</script></action>
```

Later, we can make use of the script, from our admin panel CMS pages, or directly in the template HTML files. The Nivo slider has many options that we can check here:

<http://nivo.dev7studios.com/#usage>

Keep in mind that sometimes while using the Nivo slider we can see the following error: **Stack overflow at line: 881**. If the error appears, try modifying the slider options, like using only one effect instead of `effect: 'random'`.

## See also

- ▶ *SlideDeck content slider*
- ▶ *Adding featured products to the home page*

## Magento Easy Lightbox

This time we are going to try something a bit different. Instead of coding, this time we are going to install an extension from Magento Connect, the Magento Easy Lightbox. This is a nice extension with some quite useful features.

## Getting ready

This time, as we are going to work with a Magento Connect extension instead of finding the code in the code bundle, we will need to check for the extension here:

<http://www.magentocommerce.com/magento-connect/TemplatesMaster/extension/1487/magento-easy-lightbox>

Or here:

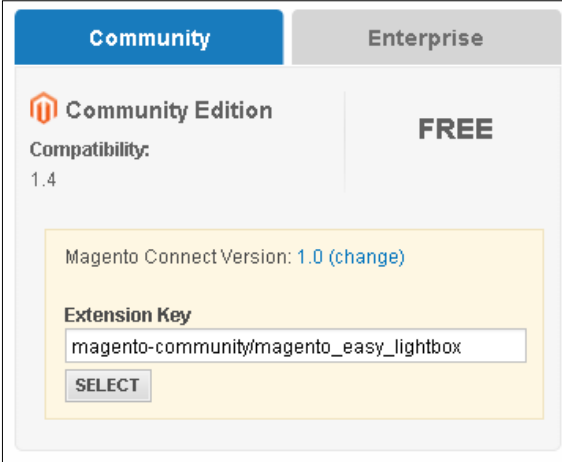
<http://bit.ly/bvtvkN>



## How to do it...

This time putting this extension to work is even simpler than in previous recipes. You only need to follow these guidelines:

1. First we need to go to the URL provided in the *Getting ready* section. Or perform a search in <http://www.magentocommerce.com/magento-connect> for Magento Easy Lightbox.
2. Once there, we need to log into our account in order to get the **Extension Key**:



Community Enterprise

Community Edition

Compatibility: 1.4

FREE

Magento Connect Version: 1.0 (change)

Extension Key

magento-community/magento\_easy\_lightbox

SELECT

3. Now that we have the extension key, it's time to log in our Magento Admin. Then go to **System | Magento Connect | Magento Connect Manager**.
4. Also log in to the **Magento Connect Manager**.
5. Introduce the **Key** in the **Install a New Extension** form:



Install New Extensions

1 Search for modules via [Magento Connect](#)

2 Paste extension key to install: magento-community/magento\_easy\_lightbox Install

6. And click on the **Install** button.
7. Once the installation is done, click on the **Log out** or the return to admin one.
8. It may also be necessary to **Log out** of the admin panel and **Log in** again.

9. Next we go to **System | Configuration** and we will be able to see a new tab called **Templates-Master**, just like in the following screenshot:



10. Now click on **Easy Lightbox**. The options panel for Easy Lightbox will appear under the **General** tab:

A screenshot of the 'Easy Lightbox' configuration panel. The panel has a title 'Easy Lightbox' in orange. Below it is a tab labeled 'General'. The configuration options are as follows:

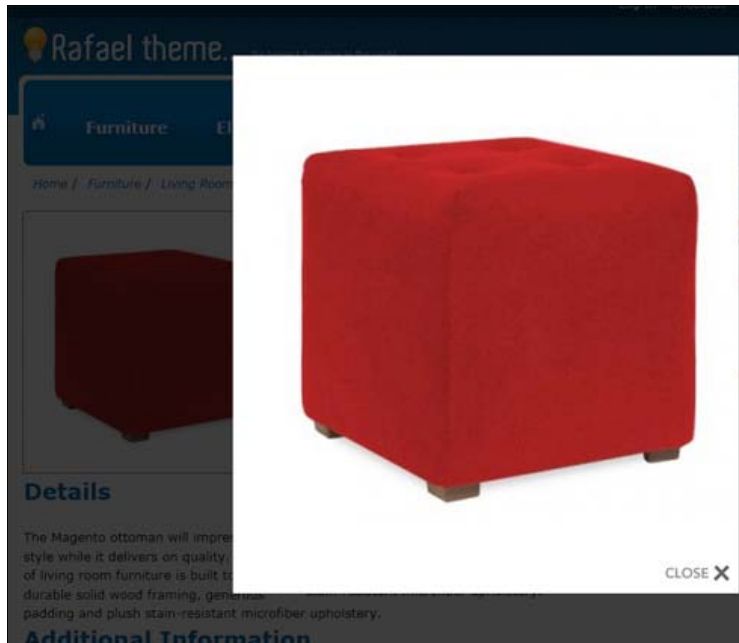
Easy Lightbox	
General	
Enable EasyLightbox	Yes
Keep Magento zoom	No
Size of main image	265_265 <small>▲ Size in pixels width_height. (Default 265_265).</small>
Size of popup image	500_500 <small>▲ Size in pixels width_height. (Default 500_500).</small>
Size of additional images	60_60

11. The extension has many interesting options, but we are going to concentrate on the main ones.
12. First, select **Yes** for **Enable EasyLightbox**.
13. Then select **No** for **Keep Magento zoom**.
14. We can also select **Yes** for **Replace empty image label with product name**.
15. With these options the extension should work, so let's go to our frontend. For example to this page: <http://127.0.0.1/magento/furniture/living-room/ottoman.html>.

16. Here we can see something similar to the following screenshot:



17. And if we click on it we should see the following popup:



18. And we are done, but feel free to play with the options available in this great extension.

## How it works...

This time it has been pretty easy to have this feature working. The only thing necessary has been to install the extension from Magento Connect and enable it from our admin panel. A quick and easy way to have a very interesting feature added to our site.

## See also

- *Using Cufon to include any font we like in our theme*

## Adding social media sharing to product page

Once we have our site ready, full of products we want to sell, wouldn't it be great to share them? This recipe is all about that. We are going to use another Magento Connect extension to add social media sharing capabilities to our site. Don't miss this one!

## Getting ready

In order to follow this recipe you will need to download the extension from here:

[http://www.magentocommerce.com/magento-connect/\\_Fluxe/extension/2333/magento-social-bookmarking](http://www.magentocommerce.com/magento-connect/_Fluxe/extension/2333/magento-social-bookmarking)

Or here:

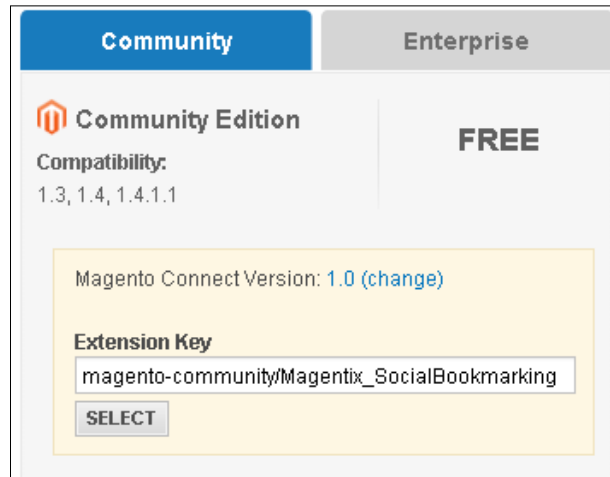
<http://bit.ly/api0Nv>

## How to do it...

This recipe involves installing a Magento Connect extension, and modifying one of our Magento template files. But don't worry, we will do this together. Let's get started:

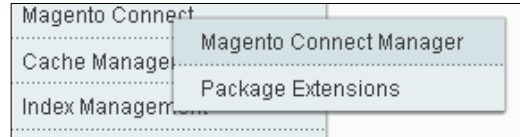
1. First we need to get the extension key for the social media sharing extension. Just navigate to the URLs suggested in the *Getting ready* section, or, make a search in the Magento Connect page for Magento Social Bookmarking Services.

2. Once there, copy the Extension Key:



The screenshot shows the Magento Community Edition interface. At the top, there are tabs for 'Community' (selected) and 'Enterprise'. Below the 'Community' tab, it says 'Community Edition' and 'Compatibility: 1.3, 1.4, 1.4.1.1'. To the right, it says 'FREE'. In the center, there is a yellow box containing the text 'Magento Connect Version: 1.0 (change)'. Below this, there is a section titled 'Extension Key' with a text input field containing 'magento-community/Magentix\_SocialBookmarking' and a 'SELECT' button.

3. Once we have the extension key we will log into our Magento admin screen, if we haven't done so yet.
4. Then we will go to the **System** menu, **Magento Connect**, and then **Magento Connect Manager**:



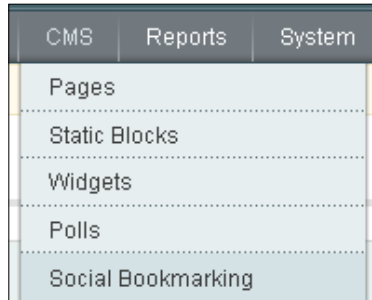
5. Log into your Magento Connect page and paste the extension key in the **Install New Extensions** form. Then click on **Install**:



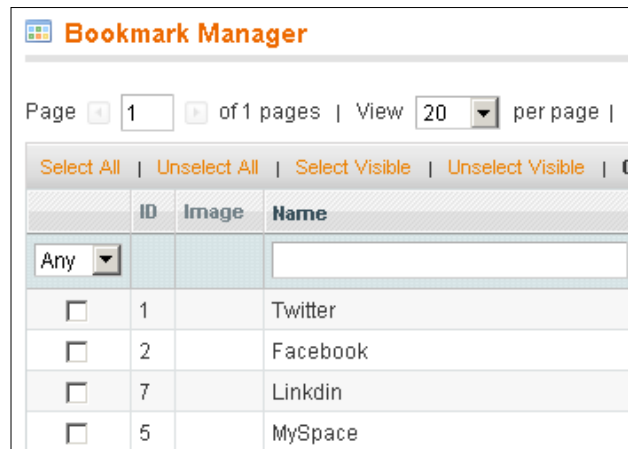
The screenshot shows the 'Install New Extensions' form on the Magento Connect page. It has two steps: 1. Search for modules via [Magento Connect](#). 2. Paste extension key to install:

6. Once the installation is done, return to the Magento admin panel.

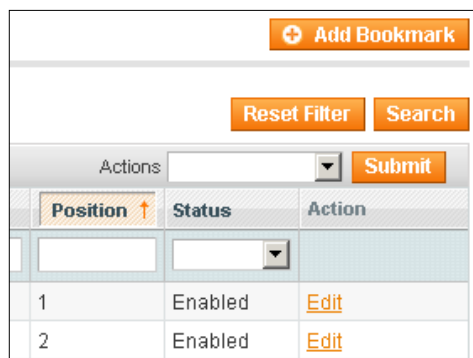
7. In the Magento admin panel go to the **CMS** menu, and then to **Social Bookmarking**:



8. This menu will direct us to the extension admin screen where we can see the default bookmarks it gives us:



9. By default they are all enabled, though the extension doesn't include icons or images for them. Don't worry, we can add them if we want. Just edit, or add, the bookmark you want:



10. The edit screen will let us add images, or change other configuration options, like the position or setting whether this book is enabled or not:

**Edit Bookmark 'Twitter'**

**Bookmark information**

Name \*

Url \*   
▲ <url> : Current page URL  
<title> : Current page Meta Title  
<bitly> : Current page Short URL (http://bit.ly)

Image

Position

Open in new window

Status

11. When we are done configuring the extension it's time to start using it. In the CMS pages it can be included as a widget, but we are going to see how to add it to our products page.
12. First we need to go to `app/design/frontend/base/default/template/catalog/product`.
13. Here, copy the `view.phtml` file.
14. Paste it in `app/design/frontend/default/rafael/template/catalog/product/view.phtml`.
15. Now open the `view.phtml` file in your favorite editor.
16. Find the following piece of code:

```
<div class="product-shop">
    <div class="product-name">
        <h1><?php echo $_helper->productAttribute($_
product, $_product->getName(), 'name') ?></h1>
```
17. And just below it add the next one:

```
<?php echo $this->getChildHtml('bookmarks') ?>
```

18. If we now load any product page, we will see something like the following screenshot:



And we are done with this recipe; don't forget to include some nice icons!

### How it works...

Like other Magento Connect extensions, usage is quite easy, though this time it was necessary to modify a template file. In general it was extremely easy to add social sharing features to our site.

### See also

- ▶ *Using Cufon to include any font we like in our theme*
- ▶ *SlideDeck content slider*
- ▶ *Nivo banner slider*

## Adding featured products to the home page

This time we are going to work on a nice feature for our site, adding featured products in the home page. Don't worry, it's not going to be that difficult.

### Getting ready

If you would like to copy the code, instead of writing it, remember you can find all the necessary code in the code bundle, in Chapter 5, recipe 4.

### How to do it...

This recipe involves some coding and some modifications in the Magento admin panel, but you can just start working:

1. First log into our Magento admin panel.
2. Then go to **CMS | Pages**.



- Click on the home page one:

Home page	home	2 columns with right bar	Main Website Main Store English French German	Enabled
-----------	------	--------------------------	---	---------

- Once inside the page editing screen select the **Content** tab.
- Click on the **Show / Hide editor** button.
- Paste the following code in the editor:

```
<div class="grid_18">
<div id="slider">  </div>
<script type="text/javascript">// <![CDATA[
    jQuery(document).ready(function($) {
        $('#slider').nivoSlider({
            effect:'sliceUp',
            slices:15,
            animSpeed:800,
            pauseTime:6000,
            directionNav:true, //Next & Prev
            directionNavHide:false, //Only show on hover
            controlNav:true, //1,2,3...
            pauseOnHover:false, //Stop animation while hovering
            captionOpacity:.7 //Universal caption opacity
        });
    });
// ]]></script>
</div>
<!-- Big banner -->
<div class="grid_6"><a href="#"></a></div>
<!-- Mini banner -->
<div class="grid_24">
<h1 class="tick">Featured toys</h1>
</div>
<!-- Big message end ->
```



Note that it's the same code we have previously seen, but without the demo featured products and footer icons.

7. Now we need to add a new category. Go to the **Catalog** menu, then **Manage Categories**. Create a new category called **Homepage**, like in the following screenshot:

✓ The category has been saved.

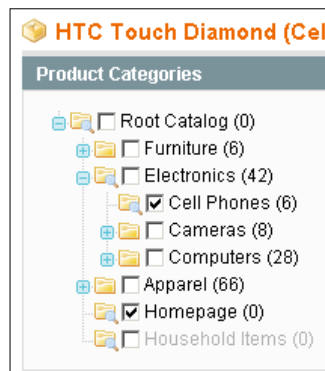
**Homepage (ID: 35)**

General Information | Display Settings | Custom

**General Information**

Name \*

8. Make sure the category is enabled, but not included in the navigation menu. Also take note of its ID, as we are going to need it later.
9. Now go to **Catalog | Manage** products.
10. Click on any product you want in order to edit it. Once inside select the **Categories** tab and add the product to the newly created **Homepage** category:



11. Now we have everything in place, our next step is to start with the code. Go to `app/design/frontend/default/rafael/template/catalog/product` and create a file called `homepage.phtml`.

12. Inside this new file place the following code:

```
<?php

$cat_id = "35";
$_productCollection = Mage::getResourceModel('catalog/product_
collection')
->addAttributeToSelect(array('name', 'price', 'small_image',
'short_description'), 'inner')
->addCategoryFilter(Mage::getModel('catalog/category')->load($cat_
id));

if(!$_productCollection->count()){
    echo $this->__('No products in the homepage category.');
```

```
}else{

?>

<div id="home-slideshow">
    <?php $i=0; foreach ($_productCollection as $_product){ ?>

        <div class="grid_6">
            <p class="product_title" style="height: 50px;"><?php
echo $_product->getName(); ?> <br /><span class="orange">$<?php
echo number_format($_product->getFinalPrice(),2);?></span></p>
            <div class="product_image">
                <div></div>
                <a href="#">htmlEscape($this->getImageLabel($_product, 'small_image')) ?>"
style="border: 4px solid #FBA826;" /></a></div>
                <a class="price" href="<?php echo $_product-
>getProductUrl() ?>" style="color: #ffffff !important">add to
cart</a>
            </div>

            <?php } ?>

        </div>

    <?php } ?>
```

13. Now that we have our template file, we need to load it in our layout file. Let's go to `app/design/frontend/default/rafael/layout` and open `page.xml`.

14. Here, find the following code:

```
<block type="core/text_list" name="right" as="right"
  translate="label">
  <label>Right column</label>
</block>
```

15. And just below it add the following:

```
<block type="core/template" name="homepage" as="homepage"
  template="catalog/product/homepage.phtml"/>
```

16. One last step awaits us; we need to modify our 2columns-right.phtml file. We will go to `app/design/frontend/default/rafael/template/page` and open `2columns-right.phtml`.

17. Here, find the following code:

```
if ($checkhome) {
?>
    <div class="grid_18">
        <?php echo $this->getChildHtml('content') ?>
    </div>
    <div class="grid_6">
        <?php echo $this->getChildHtml('right') ?>
    </div>
<?php

    }else{
        echo $this->getChildHtml('content');
    }
?>
```

18. And change it so it looks like the following:

```
if ($checkhome) {
?>
    <div class="grid_18">
        <?php echo $this->getChildHtml('content') ?>
    </div>
    <div class="grid_6">
        <?php echo $this->getChildHtml('right') ?>
    </div>
<?php

    }else{
        echo $this->getChildHtml('content');
        echo $this->getChildHtml('homepage');
    }
?>

    <div class="clear"></div><br/>
```

```

        <div class="grid_24">
            <h1 class="more_products text_right"><a
class="orange_link" href="#">Of course we have many more toys,
take a look!</a></h1>
        </div>
        <!-- Big message end -->
        <p>&nbsp;</p>
        <div class="grid_6">
            <h3 id="lock"><a class="blue_link"
href="#">Buying in our store is secure</a></h3>
        </div>
        <div class="grid_6">
            <h3 id="question"><a class="blue_link"
href="#">Got any questions? Ask us!</a></h3>
        </div>
        <div class="grid_6">
            <h3 id="shield"><a class="blue_link"
href="#">Satisfaction Guaranteed Always!</a></h3>
        </div>
        <div class="grid_6">
            <h3 id="worldwide"><a class="blue_link"
href="#">We sell our products worldwide</a></h3>
        </div>
        <!-- End of h3 messages -->
    <?php
    }
?>

```

19. If we now reload our frontend home page we will see something like the following screenshot:



20. Now we can add any product we want to the home page, in an easy way.

## How it works...

The most important part here is the `homepage.phtml` template file, but how does it work? Quite easily; first we get the products from our selected category:

```
$cat_id = "35";
$_productCollection = Mage::getResourceModel('catalog/product_
collection')
->addAttributeToSelect(array('name', 'price', 'small_image', 'short_
description'), 'inner')
->addCategoryFilter(Mage::getModel('catalog/category')->load($cat_
id));
```

Later we check if we have any results:

```
if(!$_productCollection->count()){
    echo $this->__('No products in the homepage category.');
```

```
}else{
```

And then we loop through all the results:

```
<div id="home-slideshow">
    <?php $i=0; foreach ($_productCollection as $_product){ ?>

        <div class="grid_6">
            <p class="product_title" style="height: 50px;"><?php
echo $_product->getName(); ?> <br /><span class="orange">$<?php echo
number_format($_product->getFinalPrice(), 2); ?></span></p>
```

## There's more

We can modify the `homepage.phtml` file a bit (note the code in bold):

```
<div class="grid_6">
    .
    .
    .
    <a class="price" href="http://127.0.0.1/magento/checkout/
cart/add/product/<?php echo $_product->getEntity_id() ?>/qty/1"
style="color: #ffffff !important">add to cart</a>
</div>
```

This way, clicking on the **Add to cart** link effectively adds the product to the cart.

## See also

- ▶ *Magento Easy Lightbox*
- ▶ *Adding social media sharing to the product page*

# 6

## Building Simple Extensions

In this chapter, we are going to see some recipes that will help us create some simple extensions. Though this book is not about developing Magento extensions, sometimes it will be useful to us, as a means of placing interesting content in our theme.

In the following recipes you will find an easy way to create simple extensions, and you will find yourself adding them to your projects in no time! Check the recipes:

- ▶ Installing the ModuleCreator extension
- ▶ Using the ModuleCreator extension
- ▶ Building a featured products block
- ▶ Modifying the featured products block to show the most sold products

Notice that for some of the other recipes we will make use of the first one, *Installing the ModuleCreator extension*.

### Introduction

Developing Magento extensions can be a bit tricky, but don't worry, we will be making use of the ModuleCreator extension, which will help us a lot in the process. Don't worry and let's get started!



## Installing the ModuleCreator extension

The ModuleCreator extension is a wonderful tool that will let us create basic extensions in no time. As in previous recipes, we will get this extension from the Magento connect page, with an extension key.

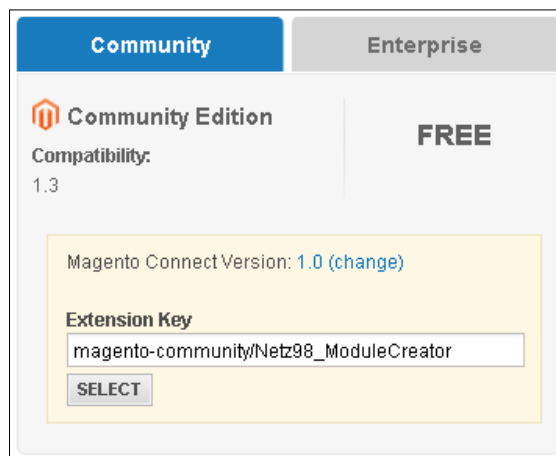
### Getting ready

It's important to follow this recipe, as other recipes in this chapter will use the base created in this recipe.

### How to do it...

Here are the steps we are going to follow to install the extension:

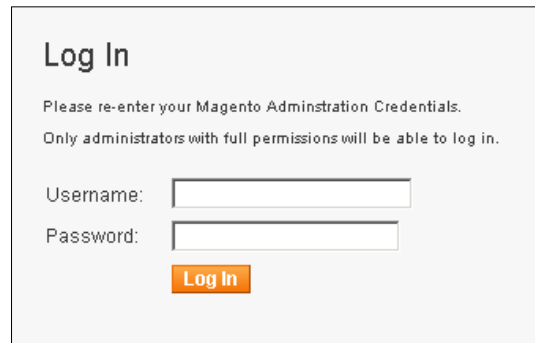
1. First we need to get the extension key, and for that we can go to this URL: <http://www.magentocommerce.com/magento-connect/Daniel+Nitz/extension/1108/modulecreator> or this one: <http://bit.ly/b4osxV>
2. In that page we can see all the details of the ModuleCreator extension, and most important of all, the extension key:



The screenshot shows the Magento Connect interface for the 'ModuleCreator' extension. At the top, there are tabs for 'Community' (selected) and 'Enterprise'. Below the tabs, the 'Community Edition' is highlighted with a 'FREE' label. The 'Compatibility' is listed as '1.3'. A yellow box contains the 'Magento Connect Version: 1.0 (change)' and the 'Extension Key' field, which is populated with 'magento-community/Netz98\_ModuleCreator'. A 'SELECT' button is located below the extension key field.

3. Once we have that **Extension Key** we will log into our Magento admin panel.
4. Here, go to **System | Magento Connect | Magento Connect Manager**.

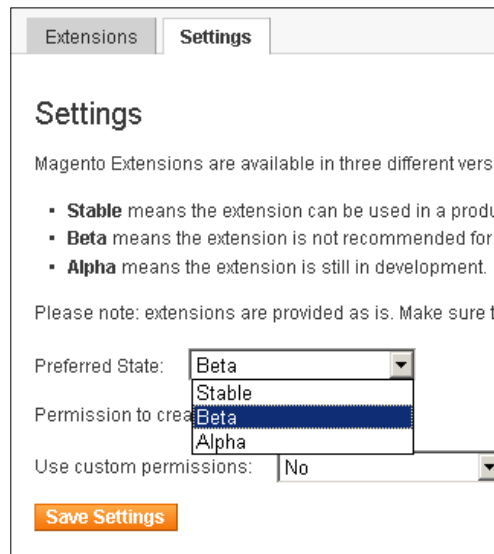
5. There you will be asked to log into the **Magento Connect Manager**; usually it's the same account as the Magento admin panel administrator. The login screen looks like the following screenshot:



The screenshot shows a 'Log In' form with the following elements:

- Log In** (Section Header)
- Please re-enter your Magento Administration Credentials.
- Only administrators with full permissions will be able to log in.
- Username:
- Password:
- Log In** (Orange button)

6. Once we are logged in, we will paste the **Extension Key** in the form, and click on the **Install** button.
7. The installation process will follow, but, in case it doesn't, maybe it has to do with our **Magento Connect Manager** settings. By default, **Magento Connect** tries to download stable extensions. But in this case, the ModuleCreator is not stable.
8. Go to the **Settings** panel, and you will see an image like the following screenshot:



The screenshot shows the 'Settings' panel with the following elements:

- Extensions** | **Settings** (Tabs)
- Settings** (Section Header)
- Magento Extensions are available in three different versions:
- **Stable** means the extension can be used in a production environment.
  - **Beta** means the extension is not recommended for production.
  - **Alpha** means the extension is still in development.
- Please note: extensions are provided as is. Make sure to test them before using them in a production environment.
- Preferred State:
- Permission to create extensions:
- Use custom permissions:
- Save Settings** (Orange button)

9. There we can select the **Preferred State** of the extensions we want to download. In this case, for the ModuleCreator, we will select **Beta**. Once done, we can save the settings.
10. Return to the **Extensions** tab, and try to install the extension again. All should go fine.
11. After the installation process is completed, click on the **Return to Admin** link that appears on the top of the page. We are done with the installation!

### How it works...

As in previous recipes in the chapter, we have installed an extension thanks to the Magento Manager panel. It's a very easy process; though this time we needed to change some settings in order for the extension to be installed.

### See also

- ▶ *Using the ModuleCreator extension*
- ▶ *Building a featured products block*
- ▶ *Modifying the featured products block to show the most sold products*

## Using the ModuleCreator extension

In the previous recipe we saw how to install the ModuleCreator extension and, in this one, we are going to see how to make use of it. As commented before, there's a quick and easy way to lay a foundation for an extension.

### Getting ready

It's important to follow this recipe, as other recipes in this chapter will use the base created in this one. Also you can find the generated code in the code bundle in Chapter 6, recipe 2.

### How to do it...

Once we have the ModuleCreator extension installed, making use of it is as easy as follows:

1. The extension will be installed in a folder called `moduleCreator`. In our case we need to go to `http://127.0.0.1/magento/moduleCreator/`.
2. There we will see a login screen that asks for our username and password. This is the same as our Magento admin panel user and password. Enter them.

- Once logged in, we will be able to see a wizard form; check out the following screenshot:

**Magento Module Creator**

Skeleton Template: Blank News Module ▼  
(you could build your own)

Namespace:  
(e.g. your Company Name)  
Packt

Module:  
(e.g. Blog, News, Forum)  
Featured

Magento Root Directory:  
(auto detected)  
C:\xampp\htdocs\magento

Design:  
(interface, default is 'default')  
default

Design:  
(theme, default is 'default')  
rafael

- How do we fill in this form? Let's talk a bit about that.
- First we need to write the **Namespace**; this will help so our extension doesn't conflict with others. Hey! We may download other extensions called **Featured**.
- Next we have to give a name to the module, and in our example it's going to be **Featured**.
- We also have to place the root directory, though in this case it has been auto detected.
- The following field is for the **Design**, the interface, and we are using **default**, so we write it.
- The last field is **Design** again, but this time for the theme. Write **rafael** here, or the theme you are using.
- The last step is to click on the **Create** button, and then watch the creation of the files.
- The ModuleCreator extension will create the following files:

```
app/etc/modules/Packt_Featured.xml
```

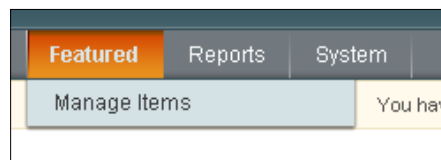
```
app/code/local/Packt/Featured/Block/Featured.php
```

```
app/code/local/Packt/Featured/controllers/IndexController.php
```

```
app/code/local/Packt/Featured/etc/config.xml
```

```
app/code/local/Packt/Featured/Model/Featured.php
app/code/local/Packt/Featured/Model/Mysql4/Featured.php
app/code/local/Packt/Featured/Model/Mysql4/Featured/Collection.php
app/code/local/Packt/Featured/Model/Status.php
app/code/local/Packt/Featured/sql/featured_setup/mysql4-install-
0.1.0.php
app/design/frontend/default/rafael/layout/featured.xml
app/design/frontend/default/rafael/template/featured/featured.
phtml
app/code/local/Packt/Featured/Block/Adminhtml/Featured.php
app/code/local/Packt/Featured/Block/Adminhtml/Featured/Edit.php
app/code/local/Packt/Featured/Block/Adminhtml/Featured/Grid.php
app/code/local/Packt/Featured/Block/Adminhtml/Featured/Edit/Form.
php
app/code/local/Packt/Featured/Block/Adminhtml/Featured/Edit/Tabs.
php
app/code/local/Packt/Featured/Block/Adminhtml/Featured/Edit/Tab/
Form.php
app/code/local/Packt/Featured/controllers/Adminhtml/
FeaturedController.php
app/code/local/Packt/Featured/Helper/Data.php
app/design/adminhtml/default/rafael/layout/featured.xml
```

12. As you can see the extension has saved us a lot of work! Creating all those files manually would take quite some time.
13. Now we can go to our Magento admin and see that we have a new menu entry, as shown in the following screenshot:



14. We are done creating our basic extension. Wasn't it easy?

## There's more...

If the extension we just created doesn't work, and you can't open the extension admin panel, don't worry. Some versions of ModuleCreator have this tiny bug but we can solve it easily.

Just open `app/code/local/Packt/Featured/controllers/Adminhtml/FeaturedController.php` and find the following piece of code:

```
public function indexAction() {
    $this->_initAction()
    ->renderLayout();
}
```

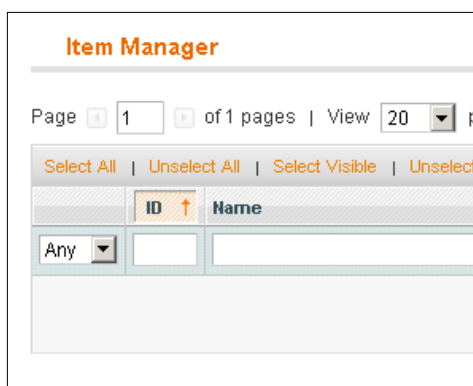
And change it to the following:

```
public function indexAction() {
    $this->_initAction()
    ->_addContent($this->getLayout()->createBlock('featured/
adminhtml_featured'))
    ->renderLayout();
}
```

And that will solve the problem! We can use the extension as we wish.

## How it works...

Once we have created all the files, and, if necessary, corrected the little bugs we have found, the extension becomes fully functional. We can use it as any other extension. Just go to the Magento admin panel, then to **Featured | Manage Items**, and you will be able to see a window similar to the following screenshot:



There you'll see a list of items, and, if we want to create a new one, just click on the **Add item** button that appears to the right. This button will lead us to a form like the one we can see in the following screenshot:

**Add Item**

**Item information**

Name \*

File

Status

Content \*

We can use this form to add as many items as we want, in quite an easy way.

## See also

- ▶ *Installing the ModuleCreator extension*
- ▶ *Building a featured products block*
- ▶ *Modifying the featured products block to show the most sold products*

## Building a featured products block

In previous recipes we have seen how to download the ModuleCreator extension, and also how to create a basic module layout with it. In this recipe, we will see how this basic module layout can be used in some interesting ways.

Along this recipe we will be using this basic layout to create a featured products block that will appear just below any product page. Check it out! I'm sure you will like it!

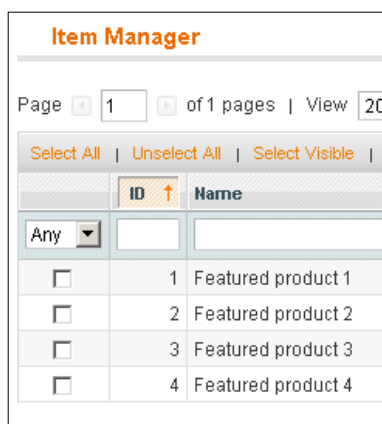
## Getting ready

You can find the code for this recipe (except for the JavaScript libraries that you can download) in the code bundle in Chapter 6, recipe 3.

## How to do it...

This recipe involves quite a few steps. So put all your attention into it, and let's start:

1. First of all we need some images to use, so try to get some. I'm going to prepare four images of 500px width and 365px height. But you can use any image/photo you like.
2. Next we will log in our Magento admin panel and open the **Featured** menu and select the **Manage items** option. Remember we did that in the previous recipe.
3. Once there, add the items you want, so we have some elements to work with. For example, I have:



The screenshot shows the 'Item Manager' interface in the Magento admin panel. It includes a header with the title 'Item Manager', a pagination bar showing 'Page 1 of 1 pages' and 'View 20', and a toolbar with buttons for 'Select All', 'Unselect All', and 'Select Visible'. Below the toolbar is a table with columns for 'ID' and 'Name'. The table contains four rows of data, each with a checkbox in the first column and the product name in the second column.

	ID	Name
<input type="checkbox"/>	1	Featured product 1
<input type="checkbox"/>	2	Featured product 2
<input type="checkbox"/>	3	Featured product 3
<input type="checkbox"/>	4	Featured product 4

4. Note that all the uploaded images can be found inside the media folder, so check there if you want to see if they have been uploaded.
5. Now to the coding part. By default the `ModuleCreator` extension puts some interesting code in the files we are about to see. But we are going to modify it, so you can see other ways of doing things. Then you can choose your favorite, or try others!
6. Open the file `app/code/local/Packt/Featured/Block/Featured.php`.
7. There look for the following method:

```
public function getFeatured()
{
    if (!$this->hasData('featured')) {
        $this->setData('featured',
            Mage::registry('featured'));
    }
    return $this->getData('featured');
}
```



8. And replace it with the following code:

```
public function getFeatured()
{
    return Mage::getModel('featured/featured') -
>getCollection();
}
```

9. Here we are making use of the featured model. We can find it in our featured extension folder, Model folder. We could place our own methods there, but now we will use the `getCollection()` one, which our model gets by extending `Mage_Core_Model_Abstract`.

10. Now, another file we have to modify is the `app/design/frontend/default/rafael/template/featured/featured.phtml`.

11. Open the file and replace all its contents with the following piece of code:

```
<h4>Featured products</h4>
<?php

    $featured = $this->getFeatured();
    foreach ($featured as $product) {
        echo $product->getTitle();
        echo "<div>";
        echo $product->getContent();
        echo "</div>";
    }

?>
```

12. We are almost there; we have our block and our PHTML file. But if we want our block to be shown in our product pages, we will need to modify a layout file.

13. Open the `app/design/frontend/default/rafael/layout/catalog.xml` file.

14. Here, look for this code:

```
<catalog_product_view translate="label">
```

15. In that XML block look for the following code:

```
<reference name="content">
```

16. At the end of this reference tag add the following highlighted coded:

```
    <block type="featured/featured" name="featured"
as="featured" template="featured/featured.phtml"/>
</reference>
<reference name="right">
```

17. Here ends the first step to get our block working, and if we go to any product page, for example: <http://127.0.0.1/magento/electronics/cell-phones/nokia-2610-phone.html>, we will be able to see something like this:



18. But there are no images. In our next step we are going to add them, so our work continues!
19. Just return to `app/design/frontend/default/rafael/template/featured/featured.phtml` and modify its `foreach` loop so its looks like the following:

```
foreach ($featured as $product) {
    echo $product->getTitle();
    echo "<br/>";
    ?>
    getTitle(); ?>" title="<?php $product->getTitle(); ?>" />
    <br/>
    <?php
    echo $product->getContent();
    echo "<br/><br/>";
}
```

20. If we reload the page, this will produce some images, as follows:



21. Now it's time to add the slider. We are going to use the great Pikachoose, which you can download from <http://pikachoose.com/>.
22. For this example I'm going to use Version 4.1.6 Stable, but any other should also be fine.
23. Download and unzip the file.
24. We are going to place some of the files included in our `skin/frontend/default/rafael` folder.
25. First check that inside the `skin/frontend/default/rafael/js` folder we have a copy of jQuery. For example, version `jquery-1.4.3.min.js`.
26. Now look inside the `pikachoose` library folder we just downloaded. There should be a folder called `assets/js`.
27. Inside there should be a file called `jquery.pikachoose.js`.
28. Copy and paste it in `skin/frontend/default/rafael/js`.
29. Now return to the downloaded folder, and inside the `assets` folder you will find a folder called `images`.
30. Copy all its content inside our `skin/frontend/default/rafael/img` folder.
31. There's one more file we need to copy from the downloaded folder; it's the one we can find inside the `bottom/css` folder, the `styles.css` file.
32. Rename it to `pikachoose.css` and paste it inside the `skin/frontend/default/rafael/css` folder.
33. Open the file.
34. Remove the following lines:

```
body {font-family: helvetica, arial, sans-serif;}
a{color:white;}
```
35. Replace all instances of the following path:

```
../../assets/images/
```
36. With this one:

```
../img/
```
37. Now we must return to edit our other files. Open `app/design/frontend/default/rafael/template/featured/featured.phtml`.
38. Modify it so it looks like the following:

```
<h4>Featured products</h4>
<ul id="pikame">
<?php
    $featured = $this->getFeatured();
```

```

        foreach ($featured as $product) {
            ?>
            <li>getTitle();
?>" title="<?php echo $product->getTitle(); ?>" /></a><span><?php
echo $product->getContent(); ?></span></li>
            <?php
        }
    ?>
</ul>
<script language="javascript">
    <!--
        jQuery(document).ready(function($) {
            $("#pikame").PikaChoose();
        });
    -->
</script>

```

39. With this done there's one last thing before we can take a look at the result of our work. Now we need to edit the file `app/design/frontend/default/rafael/layout/catalog.xml`.

40. Open the file and look for the following code:

```
<reference name="head">
```

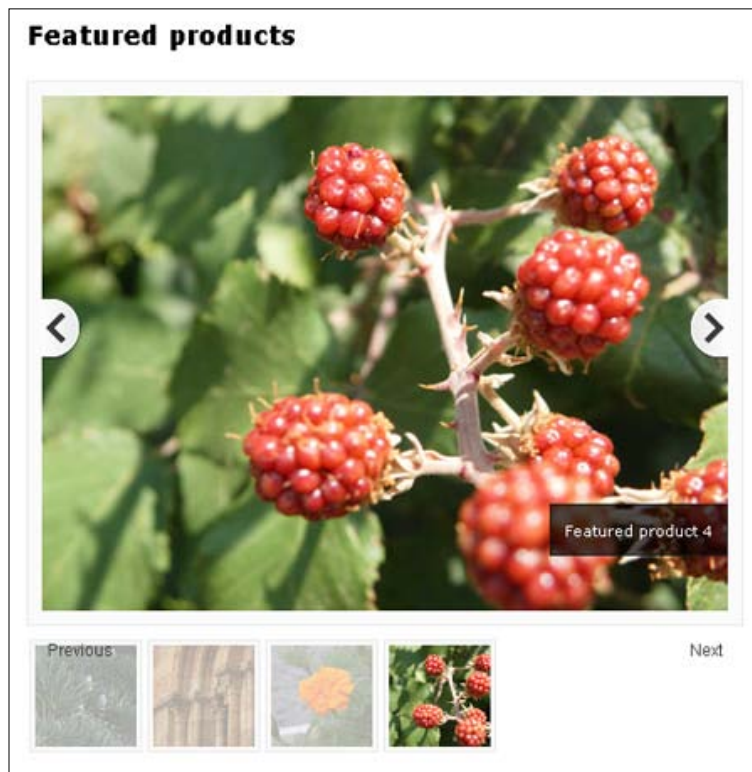
41. Add this code to the end of the block, so it looks like the following:

```

        <action method="addJs"><script>../skin/frontend/
default/rafael/js/jquery.pikachoose.js</script></action>
        <action method="addCss"><stylesheet>css/pikachoose.
css</stylesheet></action>
    </reference>

```

42. Maybe you need to edit the CSS file so it suits your needs. All our work will result in something like the following screenshot:



And we are done!

### How it works...

Thanks to the ModuleCreator extension we have been able to quickly create a featured products module and place it inside a block in each product page. Now it's up to you to extend this simple extension to suit your needs.

### See also

- ▶ *Installing the ModuleCreator extension*
- ▶ *Using the ModuleCreator extension*
- ▶ *Modifying the featured products block to show the most sold products*

## Modifying the featured products block to show the most sold products

In this recipe we will modify the products block a bit. We will see how easily we can modify the code from the previous recipe to show the most sold products, instead of the featured products we could insert from the admin panel.

### Getting ready

You can find the code for this recipe (except for the JavaScript libraries that you can download) in the code bundle in Chapter 6, recipe 4. Note that you will need the code from the previous recipe, as in this one we will only see the files we need to modify.

### How to do it...

This is an easy recipe, with only a few steps to follow:

1. First open `app/code/local/Packt/Featured/Block/Featured.php`.
2. Add the following method to the file:

```
public function getMostSold()
{
    $storeId = Mage::app()->getStore()->getId();

    $_productCollection =
Mage::getResourceModel('catalog/product_collection')
    ->addAttributeToSelect(array('name', 'price', 'small_
image', 'short_description', 'description'))
    ->setStoreId($storeId)
    ->addStoreFilter($storeId)
    ->setOrder('ordered_qty', 'desc')
    ->setPage(1, 6);

    return $_productCollection;
}
```

3. The most important part here is the `setPage`, where we can define the number of elements we want returned.
4. The next file to edit is `app/design/frontend/default/rafael/template/featured/featured.phtml`.
5. Remove all the content and place the following:

```
<h4>Most sold</h4>
<ul id="pikame">
```

```
<?php
    $featured = $this->getMostSold();
    foreach ($featured as $product) {
        ?>
        <li><a href="<?php echo $product->getProductUrl() ?>">htmlEscape($this->getImageLabel($product, 'small_image')) ?>"
title="<?php echo $this->htmlEscape($this->getImageLabel($product,
'small_image')) ?>" /></a></a><span><?php echo $product->getShort_
description(); ?></span></li>
        <?php
    }
    ?>
</ul>
<script language="javascript">
    <!--
        jQuery(document).ready(function($) {
            $("#pikame").PikaChoose();
        });
    -->
</script>
```

6. This code is quite similar to the previous one, and you will most likely remember it from previous recipes in the book. If we reload the page, we will see something like this:



7. As you can see, it is similar to the previous recipe, but this time showing products from our store's database.

### How it works...

This recipe is quite similar to the previous one, but we have seen how by only modifying two files, we can get totally different results. It's a very visual recipe thanks to the Pikachoose library.

### See also

- ▶ *Installing the ModuleCreator extension*
- ▶ *Using the ModuleCreator extension*
- ▶ *Building a featured products block*





# 7

## Localization and Other Tips

In our previous chapter we saw how to create simple extensions to show relevant info in our theme. In this one we are going to return to our theme, in order to add some features to it. It's going to be a fun chapter, with some interesting recipes such as the following:

- ▶ Localizing our theme
- ▶ Getting the current store
- ▶ How to create a multi-currency store
- ▶ How to create CMS pages
- ▶ Translating the topmenu

Interesting recipes, aren't they? Then why don't we start with the chapter? Let's go!

### Introduction

These are going to be quick and useful recipes that will help us in adding interesting features to our theme.

### Localizing our theme

Localizing our theme is an important task, so we can assure that our theme is presented in the correct language to each visitor. Also it isn't too hard to achieve. We are going to see just that in this recipe.

## Getting ready

You can find the code for this recipe in the code bundle for the book in Chapter 7, recipe 1.

## How to do it...

This is going to involve some coding, so prepare your favorite editor and follow these steps:

1. First we are going to decide which texts we need to translate, take for example this file: `app/design/frontend/default/rafael/template/page/2columns-right.phtml`.
2. In that file we can find the following piece of code:

```
<div class="grid_6">
    <h3 id="lock"><a class="blue_link"
href="#">Buying in our store is secure</a></h3>
</div>
<div class="grid_6">
    <h3 id="question"><a class="blue_link"
href="#">Got any questions? Ask us!</a></h3>
</div>
<div class="grid_6">
    <h3 id="shield"><a class="blue_link"
href="#">Satisfaction Guaranteed Always!</a></h3>
</div>
<div class="grid_6">
    <h3 id="worldwide"><a class="blue_link"
href="#">We sell our products worldwide</a></h3>
</div>
```

3. And we can change it so it looks like the following:

```
<div class="grid_6">
    <h3 id="lock"><a class="blue_link"
href="#"><?php echo $this->__('Buying in our store is secure')
?></a></h3>
</div>
<div class="grid_6">
    <h3 id="question"><a class="blue_link"
href="#"><?php echo $this->__('Got any questions? Ask us!') ?></
a></h3>
</div>
<div class="grid_6">
    <h3 id="shield"><a class="blue_link"
href="#"><?php echo $this->__('Satisfaction Guaranteed Always!')
?></a></h3>
</div>
<div class="grid_6">
    <h3 id="worldwide"><a class="blue_link"
```

```
href="#"><?php echo $this->__('We sell our products worldwide')
?></a></h3>

</div>
```

4. Here we are using the `$this->__()` method to translate the string passed as parameter. But, where's Magento going to find those translations? Easy, in the `locale` folder of our theme.
5. Go to `app/design/frontend/default/rafael/locale`. If the folder `locale` doesn't exist, create it.
6. Inside that `locale` folder we are going to create another three folders:
  - ☐ `de_DE`
  - ☐ `en_US`
  - ☐ `fr_FR`
7. Each one of these folders relates to a language. If you don't know how your language is equivalent to the symbol, an easy way to find out is by going to: <http://www.magentocommerce.com/translations> and downloading your language file. Note that the ZIP file for each language will have a same name as the folders we have just created.
8. Once we have these folders in place we need to create a file inside each one. This file will be called `translate.csv`.
9. The first file is going to be `app/design/frontend/default/rafael/locale/en_US/translate.csv`, and these are its contents:
 

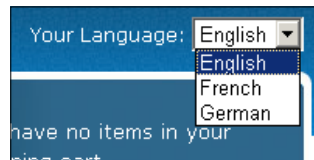
```
"Buying in our store is secure","Buying in our store is secure"
"Got any questions? Ask us!","Got any questions? Ask us!"
"Satisfaction Guaranteed Always!","Satisfaction Guaranteed Always!"
"We sell our products worldwide","We sell our products worldwide"
```
10. Note that the first parameter is the value you need to insert in the code, and the second is the translation.
11. Next we are going to edit `app/design/frontend/default/rafael/locale/fr_FR/translate.csv`:
 

```
"Buying in our store is secure","D'achat dans notre boutique est
sécurisée"
"Got any questions? Ask us!","Vous avez des questions? Demandez-
nous!"
"Satisfaction Guaranteed Always!","Satisfaction Garanti toujours!"
"We sell our products worldwide","Nous vendons nos produits à
travers le monde"
```
12. And finally edit `app/design/frontend/default/rafael/locale/de_DE/translate.csv`:
 

```
"Buying in our store is secure","Kauf in unserem Shop ist sicher"
"Got any questions? Ask us!","Haben Sie Fragen? Fragen Sie uns!"
```

"Satisfaction Guaranteed Always!","Zufriedenheit garantiert immer!"  
"We sell our products worldwide","Wir verkaufen unsere Produkte weltweit"

13. Now everything is in place, but let's try to change our site language using this dropdown:

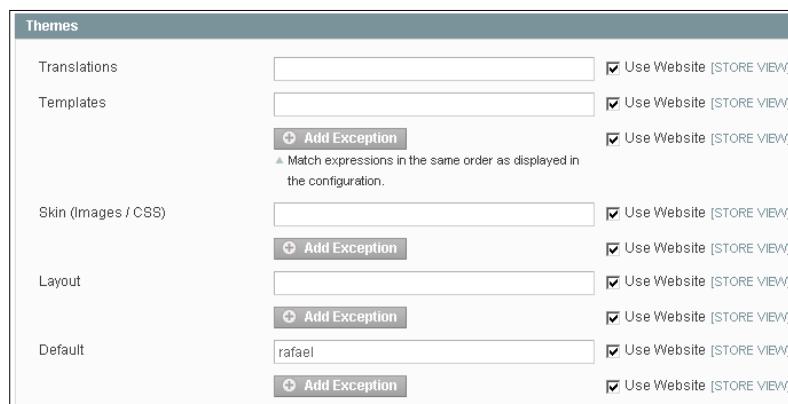


We will notice that it doesn't work. Our text doesn't translate, and even our theme is changed to the default one. But we still need to make some modifications in our Magento backend.

14. Log into your Magento Admin panel and go to **System | Configuration**.  
15. Here, we will go to the **General** tab, and select the **Design** option.  
16. Here, we can select the store we want, as can be seen in the following screenshot:



17. Here, we need to select the store (French in the example). And in the Design option we need to make sure our theme is enabled:



18. There's still one step we need to carry out, so everything works properly. Let's go to **System | Configuration | General | General**.
19. Here, select the **Current Configuration Scope** to **French**.
20. In the **Locale** tab of the **General** screen select **Locale** to **French (France)**.
21. The screen, with all these changes, should look more or less like the following:

The screenshot shows the Magento 2 Configuration interface. On the left, the 'Current Configuration Scope' is set to 'French'. Below it, there's a 'Configuration' section with 'GENERAL' selected. The main content area is titled 'General' and has two tabs: 'Countries Options' and 'Locale Options'. The 'Locale Options' tab is active, showing 'Locale' set to 'French (France)' and 'First Day of Week' set to 'Sunday'.

22. Save this configuration and return to the frontend of our site.
23. Try the language dropdown again; you should see the language changes:



And we are done!

## How it works...

It's pretty easy; now we only need to create the translate files by using the following Magento method:

```
<?php echo $this->__('Satisfaction Guaranteed Always!') ?>
```

Then configure Magento accordingly, and that's all we need. The `$this->__($string)` will search for the translation of the string in our language files. If it doesn't find it, the `$string` passed as argument will be used.

## See also

- ▶ *Getting the current store*
- ▶ *How to create a multi-currency store*
- ▶ *How to create CMS pages*
- ▶ *Translating the topmenu*

## Getting the current store

In our previous recipe we saw how to change the texts of our store. It was an easy task thanks to Magento features, but, think about the images we have in our theme. How can we change these images depending on the language being used? This is what we are about to see in this recipe.

## Getting ready

One easy way to know this is by getting the current store, as languages are linked to stores.

You can find the code for this recipe in the code bundle for the book Chapter 7, recipe 2.

## How to do it...

This is a quick recipe, so let's get started:

1. Take for example the following image of our theme:



2. Wouldn't it be great if it was translated into the current language? Let's do so.
3. First, as this piece of code is on a CMS page, and we can't run PHP there, we need to move that code to a PHTML file.
4. We need to go to our Magento Admin panel, then to **CMS | Pages**. There open the **Home page** one.

5. Cut all its code and leave this page blank. Save it.
6. Open the file `app/design/frontend/default/rafael/template/catalog/product/homepage.phtml`.
7. Paste the code we have just cut, on top of all previous code:

```
<div class="grid_18">
<div id="slider">  </div>
<script type="text/javascript">// <![CDATA[
    jQuery(document).ready(function($) {
        $('#slider').nivoSlider({
            effect:'sliceUp',
            slices:15,
            animSpeed:800,
            pauseTime:6000,
            directionNav:true, //Next & Prev
            directionNavHide:false, //Only show on hover
            controlNav:true, //1,2,3...
            pauseOnHover:false, //Stop animation while hovering
            captionOpacity:.7 //Universal caption opacity

        });

    });
// ]]></script>
</div>
<!-- Big banner -->
<div class="grid_6"><a href="#"></a></div>
<!-- Mini banner -->
<div class="grid_24">
<h1 class="tick">Featured toys</h1>
</div>
<!-- Big message end ->
```

8. This code won't work as it is, as there are some CMS pages code, such as `{{skin}}`. We need to replace those, so that code looks like the following:

```
<div class="grid_18">
<div id="slider"> <img title="Banner 2"
```



```
src="<?php echo Mage::getBaseUrl(); ?>/skin/frontend/default/
rafael/img_products/big_banner.jpg" alt="Big banner 2" /> </div>
<script type="text/javascript">// <![CDATA[
    jQuery(document).ready(function($) {
        $('#slider').nivoSlider({
            effect:'sliceUp',
            slices:15,
            animSpeed:800,
            pauseTime:6000,
            directionNav:true, //Next & Prev
            directionNavHide:false, //Only show on hover
            controlNav:true, //1,2,3...
            pauseOnHover:false, //Stop animation while hovering
            captionOpacity:.7 //Universal caption opacity

        });

    });
// ]]></script>
</div>
<!-- Big banner -->
<div class="grid_6">
    <a href="#"></a></div>
<!-- Mini banner -->
<div class="grid_24">
<h1 class="tick">Featured toys</h1>
</div>
<!-- Big message end ->
```

9. Now we have all the pieces in place, what's next? Just look for the following piece of code:

```
<div class="grid_6">
    <a href="#"></a></div>
```

10. And replace it so it looks like the next block:

```
<div class="grid_6">

    <?php

        $store = Mage::app()->getStore();
        $name = $store->getName();

        switch($name) {

            case 'German':
                ?>
                <a href="#"></a>
                <?php
                break;

            case 'French':
                ?>
                <a href="#"></a>
                <?php
                break;

            default:
                ?>
                <a href="#"></a>
                <?php

            }
        ?>
    </div>
```

11. Here we are getting the store, and later its name, in order to know which store we are in. Later, we will be showing one image or another, so if we were in the German store we should see something like:



12. This is a quick and easy way to have each image presented to the corresponding language.

### How it works...

This is a quick and easy-to-apply recipe. We only need to get the current store:

```
 Mage::app()->getStore();
```

Then its name:

```
 $store->getName();
```

And, with that, decide which image to show. And that's all.

### See also

- ▶ *Localizing our theme*
- ▶ *How to create a multicurrency store*
- ▶ *How to create CMS pages*
- ▶ *Translating the topmenu*

## How to create a multi-currency store

In previous recipes we learnt how to create a multi language store. Now, we are going to do the same with currencies. This is also very important, as visitors from other countries would like to see the store not only in their own language, but also in their currency. That would create a better user experience. In this recipe we are going to see how to do that.

## Getting ready

You can find the code for this recipe in the code bundle for the book in Chapter 7 recipe 3.

## How to do it...

As in previous recipes, this is going to be a mix of coding and configuration, so log into your Magento admin panel and follow these steps:

1. Once logged in, go to **System | configuration**.
2. Then go to **General | Currency** setup.
3. Now open the **Currency** options tab. You will see a screen similar to the following screenshot:

**Currency Options**

Base Currency: US Dollar  
▲ Base currency is used for all online payment transactions. Scope is defined by the catalog price scope ("Catalog" > "Price" > "Catalog Price Scope").

Default Display Currency: US Dollar

Allowed Currencies:

- US Dollar
- Ugandan Shilling
- Ukrainian Hryvnia
- United Arab Emirates Dirham
- Uruguayan Peso
- Uzbekistan Som
- Vanuatu Vatu
- Venezuelan Bolívar
- Venezuelan Bolívar Fuerte
- Vietnamese Dong

4. Here we can select the base currency of our store, and the default display currency. Most importantly, we can select all the allowed currencies for our store.
5. Anyway, we aren't ready yet to show our currencies in our theme. We still need to go to **System | Manage Currency Rates**.
6. Here we will see a screen like the following:

**Manage Currency Rates**

Import Service: Webservice

From Currency	To Currency	Rate
USD	EUR	1.0000

7. In that screen we need to click on the **Import** button, so the conversion rates are correctly fetched. This allows the site to convert quantities between currencies.
8. Note that without this configuration, currencies won't be shown in our theme, so don't forget to save this screen:

From	To	Rate
USD	EUR	1.0000

9. Now open `app/design/frontend/default/rafael/layout/page.xml`.
10. Look for the following piece of code:

```
<block type="page/template_links" name="footer_links"
as="footer_links" template="page/template/links.phtml"/>

</block>
```

11. Just above that add the following block:

```
<block type="directory/currency" name="currency"
template="directory/currency.phtml"/>
```

12. Now we need to open the following file `app/design/frontend/default/rafael/template/page/2columns-right.phtml`.

13. Find the following code:

```
<div id="footer">
    <div class="container_24">

        <div class="grid_6">
            <br/>
            <ul>
                <li><a href="#">About Us</a></li>
                <li><a href="#">Customer Service</a></li>
```

14. And add the following below:

```
<li><?php echo $this->getChildHtml('currency') ?></li>
```

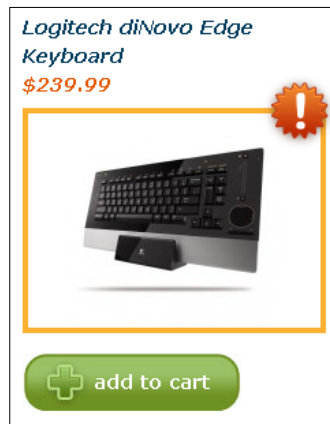
15. Now, if we reload our page, in the footer we will see something like the following:



16. With this done, we can change between currencies, and our products will reflect that, as we can see in the following product detail page:



17. Catalog pages will also reflect the currency we have selected, and enabled. We don't have to do anything; Magento will take care of everything.
18. But what about other places of our site, for example the homepage? There we have something like the following:



19. This price won't change, even if we change the current currency with the dropdown. Let's check why this is not working.
20. First we need to open `app/design/frontend/default/rafael/template/catalog/product/homepage.phtml`.
21. Here, find the following piece of code:

```
<div class="grid_6">
    <p class="product_title" style="height: 50px;"><?php
    echo $_product->getName(); ?> <br /><span class="orange">$<?php
    echo number_format($_product->getFinalPrice(),2);?></span>
```

22. In that code we were showing the final price, but it's not in the current currency. So we need to change it so it looks like the following:

```
<div class="grid_6">
    <p class="product_title" style="height: 50px;"><?php
echo $_product->getName(); ?> <br /><span class="orange">
    <?php
        $price = $_product->getFinalPrice();
        $base_currency_code = Mage::app()->getStore()-
>getBaseCurrencyCode();
        $current_currency_code = Mage::app()->getStore()-
>getCurrentCurrencyCode();
        $conversion = Mage::helper('directory')-
>currencyConvert($price, $base_currency_code, $current_currency_
code);

        echo Mage::helper('core')->currency($conversion);

    ?>
</span>
```

23. Now the price will be shown correctly, as we can see in the next image:



And we are done with this recipe!

## How it works...

In this recipe we added multi-currency features to our site. The process only required:

- ▶ The correct configuration in the Magento admin panel
- ▶ Loading of the necessary blocks in our layout and template files

- ▶ We still needed to do some tweaks to our theme, but these were easily achieved by:

- ▶ *Getting the base currency:*

```
Mage::app()->getStore()->getBaseCurrencyCode();
```

- ▶ *Getting the current currency:*

```
Mage::app()->getStore()->getCurrentCurrencyCode();
```

- ▶ *Converting the price:*

```
Mage::helper('directory')->currencyConvert($price, $base_currency_code, $current_currency_code);
```

And that's all we need to have a full multi-currency store!

### See also

- ▶ *Localizing our theme*
- ▶ *Getting the current store*
- ▶ *How to create CMS pages*
- ▶ *Translating the topmenu*

## How to create CMS pages

CMS pages can be quite useful for us to add contact info, FAQs, or other important information about our store. In this recipe, we are going to see how to add CMS pages, one for each language, and link them correctly. Let's take a look.

### Getting ready

You can find the code for this recipe in the code bundle for the book in Chapter 7, recipe 4.

### How to do it...

This is a quick recipe, so let's get started:

1. First we need need to log into our Magento admin panel.
2. Then we need to go to **CMS | Pages**.
3. There click on the **Add New Page** button.



4. We will see a page similar to the following screenshot:

5. In the page information tab, we will need to fill all the fields: the **Page Title**, the **URL Key**, and the **Store View**, in which the CMS page should be enabled.
6. Remember to also fill the content for the CMS page and save it.
7. We will need a page for each language, like so:

About Us	about-us-english	1 column
About Us	about-us-french	1 column
About Us	about-us-german	1 column

8. The URL key, second column, is the data we need to create the links.
9. Now to the coding part. Let's open `app/design/frontend/default/rafael/template/page/2columns-right.phtml`.
10. There we are going to look for the following:

```
<ul>
    <li><a href="#">About Us</a></li>
    <li><a href="#">Customer Service</a></li>
    <li><?php echo $this->getChildHtml('currency')
?></li>
</ul>
```

11. We will replace that code with the following:

```
<ul>
    <li><?php
        $store = Mage::app()->getStore();
        $name = $store->getName();
```

```

switch($name) {

    case 'German':
        ?>
        <a href="<?php echo $this-
>getUrl("about-us-german"); ?>"><?php echo $this->__('About us!')
?></a>

        <?php
        break;

    case 'French':
        ?>
        <a href="<?php echo $this-
>getUrl("about-us-french"); ?>"><?php echo $this->__('About us!')
?></a>

        <?php
        break;

    default:
        ?>
        <a href="<?php echo $this-
>getUrl("about-us-english"); ?>"><?php echo $this->__('About us!')
?></a>

        <?php

    }

    ?>
</li>
<li><a href="#">Customer Service</a></li>
<li><?php echo $this->getChildHtml('currency')
?></li>

</ul>

```

12. And we are done. With this code we will create a link for each one of the CMS pages, just the correct one for the active language.

### How it works...

We have created the necessary CMS pages in our Magento admin panel and then edited our template file so we can present the correct link. The one difference here has been the way to generate the link, this time the `getUrl`:

```
<?php echo $this->getUrl("about-us-english"); ?>
```

And that's all we need.

## See also

- ▶ *Localizing our theme*
- ▶ *Getting the current store*
- ▶ *How to create a multi-currency store*
- ▶ *Translating the topmenu*

## Translating the topmenu

Until now we have done great work localizing our theme, seeing how to achieve such things in previous recipes. But there's still something to do, our top menu. Let's take a look at it:



That's yet to be translated, and that's just what we are going to see how to do in this recipe.

## Getting ready

You can find the code for this recipe in the code bundle for the book in Chapter 7, recipe 5.

## How to do it...

There are quite a lot of things to do to in order to achieve the translation of the topmenu, so don't waste any time:

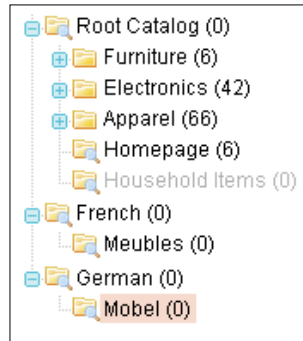
1. First we need to log into our Magento admin panel.
2. Here, go to **Catalog** | **Manage** categories.
3. We need to add two new Root Categories by clicking on the **Add Root Category** button:

A rectangular button with an orange gradient background. It features a white plus icon in a circle on the left and the text "Add Root Category" in white.

4. Make sure the category is active and save it.
5. We will also need to add subcategories to these root categories by simply clicking on the **Add Subcategory** button:

A rectangular button with an orange gradient background. It features a white plus icon in a circle on the left and the text "Add Subcategory" in white.

6. For example, create the **Furniture** category and save it. Later we will add products to these categories with the correct language to each product.
7. If everything is fine, our structure should look like this:



8. There will be three root categories, and as many subcategories as we need.
9. We are not going to see how to add products to these categories. Just remember that, at this point, we should add them to the correct categories, so each root category has the same products, each in its correct language.
10. With this done we will go to **System | Manage Stores**.
11. It was here that we created the store views, one for each language. This time we are going to work a bit differently.
12. I've deleted all the store views but the English one. However, you can rename them; just remember to, be very careful if trying this on a live site.
13. Good, now click on the **Create Store** button:



14. Here we need to define the **Website**, Main Website for our example, then the **Name**, **German** or anything you like, and the **Root Category**, which should equal the name, so they are easier to distinguish:

Store Information	
Website *	Main Website
Name *	German
Root Category *	German

15. Save the store.

16. Now we need to create a store view for this store. Just click on the **Create Store View** button:



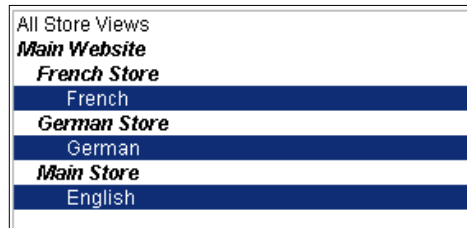
17. In the screen there, fill all fields as shown in the following image:

Store View Information	
Store *	German Store
Name *	German
Code *	german
Status *	Enabled
Sort Order	

18. There we must select the **Store**, German Store for example. We should give the store view a **Name**, like German, for example, and a **Code**, German again. And of course enable it.
19. We will do this for the other languages too.
20. Now we need to do something more. Let's go to **System | Configuration**.
21. There, look for **Current Configuration Scope**. Open the select box and select the German store.
22. Next go to the **General** tab and click on the **Design** option.
23. There make sure the rafael theme is selected in the **Themes** tab, which is the default option, as you can see in the following screenshot:

Themes		
Translations		<input checked="" type="checkbox"/> Use Website [STORE VIEW]
Templates		<input checked="" type="checkbox"/> Use Website [STORE VIEW]
	<input type="button" value="Add Exception"/>	<input checked="" type="checkbox"/> Use Website [STORE VIEW]
	▲ Match expressions in the same order as displayed in the configuration.	
Skin (Images / CSS)		<input checked="" type="checkbox"/> Use Website [STORE VIEW]
	<input type="button" value="Add Exception"/>	<input checked="" type="checkbox"/> Use Website [STORE VIEW]
Layout		<input checked="" type="checkbox"/> Use Website [STORE VIEW]
	<input type="button" value="Add Exception"/>	<input checked="" type="checkbox"/> Use Website [STORE VIEW]
Default	rafael	<input checked="" type="checkbox"/> Use Website [STORE VIEW]
	<input type="button" value="Add Exception"/>	<input checked="" type="checkbox"/> Use Website [STORE VIEW]

24. Save the configuration
25. Repeat for each language you want.
26. There's still one more change to do. Go to **CMS | Pages**.
27. There, select the enabled home page, and make sure the page is selected for all store views, like this:



28. Also, don't forget to go to **System | Configuration | General | General**.
29. There, select the **Current Configuration Scope** as **French Store**.
30. In the Locale tab of the **General** screen select **Locale** as **French (France)**.
31. Do the same for all required languages.
32. Now if we return to the front page and reload, we might need to remove the cookies first. We will see that our languages dropdown won't appear.
33. It was attached to store views, and now our default store only has one store view, English. Now that we have three stores, couldn't we have a store selector?
34. Now, to the coding part. Open `app/design/frontend/default/rafael/layout/page.xml`.
35. Find the following piece of code:
 

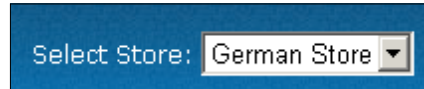
```
<block type="page/switch" name="store_language" as="store_language" template="page/switch/languages.phtml"/>
```
36. Comment, or remove it, and place the following instead:
 

```
<block type="page/switch" name="store_switcher" as="store_switcher" template="page/switch/stores.phtml"/>
```
37. Now open `app/design/frontend/default/rafael/template/page/2columns-right.phtml`.
38. Search for the store language code:
 

```
<?php echo $this->getChildHtml('store_language') ?>
```
39. Comment or remove this code and change it to:
 

```
<?php echo $this->getChildHtml('store_switcher') ?>
```

40. Now we can reload our home page and see the result:



41. And everything works. The previous code we have made for the other recipes, will still work.

42. If we wanted to edit this dropdown, to change the text, or anything, we could copy the `app/design/frontend/base/default/template/page/switch/stores.phtml` file and paste it into our own theme.

43. And that's all; we are done.

### How it works...

This long recipe relies much on admin panel configuration, and a bit of coding—just to add the necessary block and template file. It's long, but easy to achieve, and with this our site can be fully localized.

### See also

- ▶ *Localizing our theme*
- ▶ *Getting the current store*
- ▶ *How to create a multi-currency store*
- ▶ *How to create CMS pages*

# 8

## Selling Our Theme

Well, this is our last chapter; in the previous ones we have seen many recipes that could help us in setting up our theme. In this last one we are going to see some quick recipes, like packing our theme and where to sell it. These are the recipes we will look at:

- ▶ Packing our theme
- ▶ Nice features to include in our theme
- ▶ Where to sell our theme
- ▶ Where to go from here

These can be seen more as advice than anything else; anyway I really hope they are helpful to you.

### Introduction

In this last chapter we are going to see some ideas, sites, features, and tools that can give you some help while developing your Magento themes. These recipes will surely give you some inspiration in your future work. Enjoy them and also take a look at the suggested links; they are very useful!

### Packing our theme

Now our theme has everything ready, and of course we could add more features to it, or change the existing ones. But what if we want to install the theme in another Magento installation, share it, or even sell it?

We can easily achieve this by packing our theme files into a ZIP file, so let's see how.



## Getting ready

You can find the code for this recipe in the code bundle for the book in Chapter 8, recipe 1.

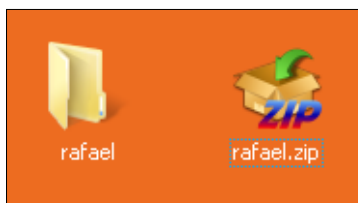
## How to do it...

Get ready and follow these steps!

1. First we are going to create a folder for our theme, where you want, like on your desktop. For example I'm going to create a `rafael` folder.
2. Inside, create the following structure:

```
→ skin
  → frontend
    → default
      → raphael
        → css
        → img
        → img_products
        → js
  → app
    → design
      → frontend
        → default
          → raphael
            → layout
            → locale
            → template
      → adminhtml
        → default
          → raphael
            → layout
    → code
      → local
        → Packt
          → Featured
            → controllers
            → etc
            → Helper
            → Model
            → sql
            → Block
    → etc
      → modules
        → Packt_Featured.xml
```

3. Of course, all these folders will have all our theme files inside them.
4. Once this is done we can pack all the folders into a ZIP file, ready for sharing:



5. Now, if we want to install the theme in another Magento installation, it would be as easy as unzipping the files and pasting them on the new Magento installation.
6. Of course, if configuration is required like selecting the homepage in the CMS menu, correctly configuring the location in the System panel, and so on, these should be included in the ZIP file.

## How it works

This is quite easy; we only need to pack all the files of our theme, and we are ready to share or sell it. There's no need for an installer, just to copy the files inside the new Magento installation. However, placing a good help file inside the ZIP file would be a great idea.

## See also

- ▶ *Where to sell our theme*
- ▶ *Where to go from here*

## Nice features for our theme

In this recipe we are going to see which features would be useful for our theme, like slideshows, menus, and many more. These examples can serve as a good inspiration when you work in your themes. Let's take a look at the following examples!

## Getting ready

This recipe is mostly about ideas of how, or where to sell our theme. So there's no code in the code bundle. Just follow the recipe steps.

## How to do it...

Proceed with the following steps and enjoy!

1. One of the most useful features are huge drop-down menus, as can be seen in the following screenshot:

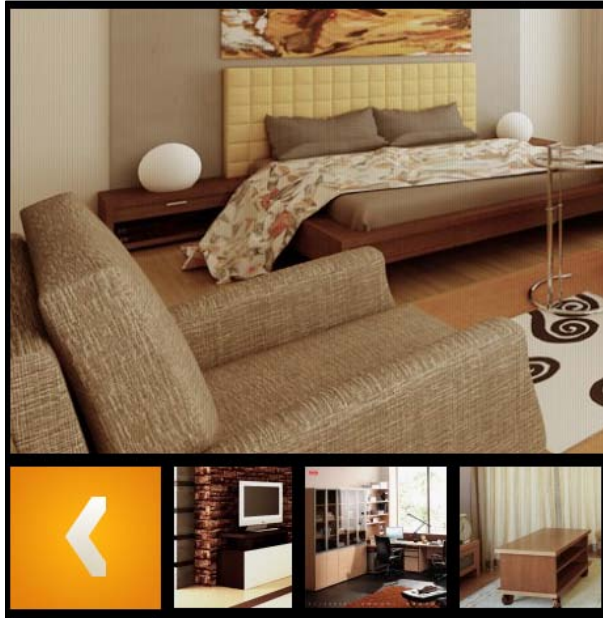
HOME	KIDS	MEN	WOMEN
Cap Sleeve		Long Sleeve	Short Sleeve
Animal		Music	Animal
Fish		Pets	Slogans
			Vintage

2. These are great for organizing our categories in a very visual way, especially if we have lots of categories.
3. Another thing we can add in our themes is a way of marking news items. Let's take a look at the following screenshot:

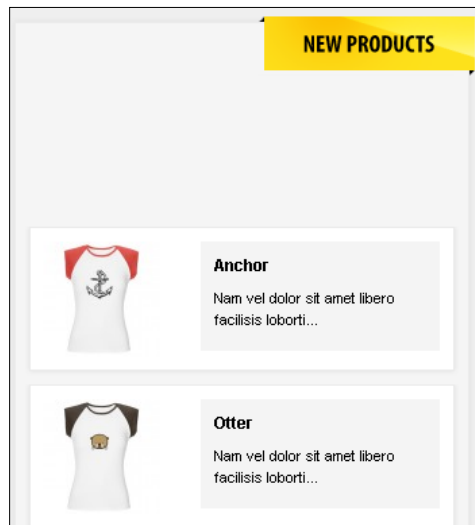


4. See how nice it is? It's a good way for promoted products to stand out over the rest. Also, note how well it shows the discounted price.

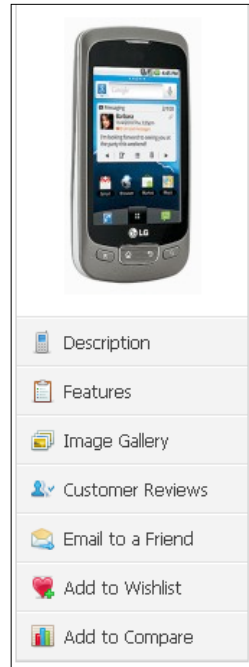
5. Another great addition to our theme could be an image gallery. Check out the following example:



6. This feature can be used not only for showing promoted products, but also to show offers, adds for Christmas campaigns, or anything you like.
7. Another one of my favorites is the product ticker. Check out this example in the following screenshot:



8. This is a very attractive way to show our site's featured products. The animation will also help draw the attention of our visitors towards certain products we want highlighted.
9. The last example is a very innovative way of organizing our product detail contents and showing them through a menu. Take a look at the following screenshot:



## How it works

Nowadays most themes include a lot of features; in this recipe we have seen some of them. Add some of them to your themes to spice them up!

The themes from which the screenshots were taken are as follows:

- ▶ <http://themeforest.net/item/tshirts-magento/241651>
- ▶ <http://themeforest.net/item/apparel-store-magento-theme/175043>
- ▶ <http://themeforest.net/item/furniture-store/123693>
- ▶ <http://themeforest.net/item/mobile-magento-theme/231559>

## See also

- ▶ *Packing our theme*
- ▶ *Where to go from here*

## Where to sell our theme

Once we have our theme ready, it is time to try to get the maximum profit from our work. In this recipe we are going to think about places, and our ideas to sell our theme.

### Getting ready

This recipe is mostly about ideas of how or where to sell our theme. So there's no code in the code bundle. Just follow the recipe steps.

### How to do it...

Let's check out the following ideas together:

1. The first place that might come to mind, where we can share or sell our theme is, of course, Magento Connect: <http://www.magentocommerce.com/magento-connect>.



2. This place has over 1455 elements on the Design and themes category, so it can be a very competitive place. Anyway, it can be an interesting place for our theme to be placed.
3. Another interesting place where we can sell our theme is on: <http://themeforest.net/>.



4. Here there are only 34 elements, at the time of writing this, on the Magento category. That's a lot less competitive, but themeforest standards are very high, so your theme should have not only a great looking design, but a good code and good documentation.
5. Also, it's a good place where you can buy themes from which you can learn some interesting techniques. A great place overall.
6. There are other sites like themeforest, for example: <http://www.mojo-themes.com>.



7. This marketplace has fewer Magento items than the previous two, so it's an interesting place where we can try to sell our Magento theme. Maybe it's not as well known as the other two, but that can help us to sell our themes in a less competitive place.
8. Let's look at yet another interesting place: <http://sitepointmarket.com/>.



9. This is a new place and for now, at the time of writing this, we can't find Magento themes here. But they are selling web themes and WordPress themes, so maybe you should try selling Magento themes here.
10. If you know of some fellow developers and designers, it would be a nice idea to create a site where you all can sell your work. Imagine sites like the following: <http://www.magthemes.com/>.



11. Or this one: <http://www.woothemes.com/why-magento/>.



12. Again the themes you can find in these places can teach you techniques, or give ideas for your own themes.
13. Another way to promote your theme would be to create a tutorial about how to do it. Of course, don't do this with your premium themes, but maybe just with your first ones, or the ones you will be using for learning purposes. A good place where you can send your tutorials is: <http://net.tutsplus.com/>.



14. These are just a few places to start, and hopefully they are useful to you!

## How it works

In this recipe we have seen some places where we can start looking to sell our theme. Learn more or do a bit of marketing of your themes. Magento themes are a bit more common these days, but still not as common as WordPress themes, so selling them can prove to be very profitable.

## See also

- ▶ *Packing our theme*
- ▶ *Where to go from here*

## Where to go from here

In our previous recipes we have seen how to pack our theme, and some places where we can try to sell it. In this recipe I will try to give you some ideas you can take a look at. I really hope these recipes are useful to you.

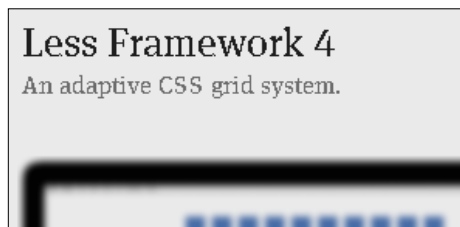
## Getting ready

This recipe is mostly about ideas of how, or where, to sell our theme. So there's no code in the code bundle. Just follow the recipe steps.

## How to do it...

Let's check out the following ideas together:

1. Use the Less Framework with your theme: <http://lessframework.com/>.



2. This framework is a CSS grid system that can adapt to the layout of our site depending on where it is being watched. This way we can provide our theme with different CSS files. This way our theme will target more devices in a correct way—an interesting tool to take into account.



3. Try a framework like the SilverFramework: <http://www.silverframework.com/>.



4. This is a nice Magento theme that can be used as a starting point for your themes. It also includes some interesting features, like an admin module to help configure some of the features of the theme. It has SEO optimization and many color schemes to choose from. Worth taking a look at!
5. Another theme that can be used as a starting point is the Yoast Blank SEO theme: <http://www.magentocommerce.com/magento-connect/Yoast/extension/974/yoast-blank-seo-theme>.



6. This is a theme that can be easily modified to satisfy our needs, and comes with an interesting number of features, like being SEO optimized, and having an admin module to configure the template. Yoast also provides SEO advices here: <http://yoast.com/articles/magento-seo/>.
7. Also there's an interesting Magento Framework yet to come: <http://diymage.com/>.



8. You can check their page to learn more, and keep an eye out for when it's being released. The features it will have are being able to drag-and-drop Magento blocks into position and change the page layout. It also works for any theme and much more. Take a look at it!

9. Another interesting tool to keep track of is Magento Go:  
<http://www.magentocommerce.com/go/>.



10. This is a new service that can let us have a Magento installation working as a service, just for a monthly fee. With many features, this is really worth taking a look at if we want to quickly have a Magento store working. It is configurable and capable of using themes. Why don't you try to create themes for Magento Go?

### How it works

In this recipe we have seen some ideas of upcoming tools that can help us in our Magento development. I hope you've enjoyed them!

### See also

- *Packing our theme*
- *Where to sell our theme*



# Quick Summary

Throughout this book we have seen some recipes that can be very useful to quickly edit some features of our site. Those recipes mostly consist of editing some files, adding our own code, or making changes to the configuration of our site.

However, there are many ways in which those modifications can be achieved, and in this chapter I want to include some recipes that show you, the reader, some different ways to achieve some tasks.

This chapter is also going to be a small summary that you can follow to get started in modifying a Magento installation. Think of this chapter as a quick start guide, or a summary you can follow if you have forgotten something.

Get ready for the following recipes:

- ▶ Step 1—laying the foundation
- ▶ Step 2—small modifications that can be done in the admin panel
- ▶ Step 3—small modifications that can be done in layout files

## Introduction

Before starting with the recipes we are going to prepare a bit of our installation. We are going to work with the default template. In fact, we are going to work as if we have just installed Magento; thus we will be able to see how to modify it effectively after a fresh installation.

This would be a good time to create a fresh Magento installation if you haven't done so already.

## Step 1—laying the foundation

For this recipe we are going to work with the default theme, just as if we have downloaded and installed Magento. Imagine we want to modify a fresh Magento installation. Let's do it.

### Getting ready

If you have followed the previous chapters in this book, you might want to create a fresh Magento installation. This way it will be easier to follow as the other changes won't mess with the ones in this recipe.

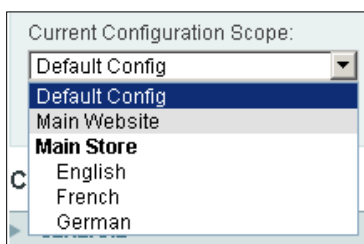
### How to do it...

Let's get started with this recipe:

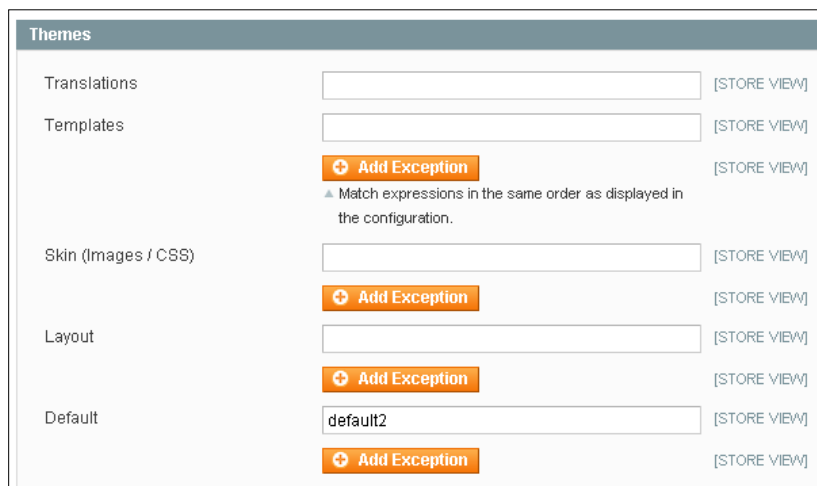
1. First open your Magento installation folder and go to `app/design/frontend/default`. There we can see all the themes that are already installed in our Magento store.
2. We are going to use the default theme, and, as it's best not to modify the original template we are going to duplicate it. Duplicate the default theme and create one called `default2`.
3. After this is done we also need to duplicate the `skin` folder. Let's do that by going to `skin/frontend/default`. There you will see another folder called `default`, and just as before duplicate it to a folder called `default2`.
4. Now log into your Magento admin panel and go to the **System** menu, then to **Cache Management**.
5. Here make sure all cache is disabled, so our modifications can be seen without issue; the last column of this screen must look like the following:

Flush Magento Cache	
Flush Cache Storage	
Actions Refresh Submit	
Associated Tags	Status
CONFIG	DISABLED
LAYOUT_GENERAL_CACHE_TAG	DISABLED
BLOCK_HTML	DISABLED
TRANSLATE	DISABLED
COLLECTION_DATA	DISABLED
EAV	DISABLED
CONFIG_API	DISABLED

6. Also don't forget to click the flush cache buttons, so if previous cache was present we get to remove it.
7. Good, now go to **System | Configuration**.
8. Here make sure you select **Default Config** in the **Configuration Scope**, as seen in the following screenshot:



9. Now, in the **General** tab, select the **Design** option. Here navigate to the **Themes** section, as you can see in the next screenshot:



10. Here, in the default input we have to insert the theme we want to use, `default2` in this example. After this is done click on the **Save Config** button.
11. We can now go to our site's frontend and refresh it to see our selected theme load.

## How it works

In this first step we have just selected the theme we want to use. This is done following the **System | Configuration | General | Design**. There we can select the theme to use and start modifying it.

## See also

- ▶ *Step 2, small modifications that can be done in admin panel*
- ▶ *Step 3, small modifications that can be done in layout files*

## Step 2—small modifications that can be done in admin panel

In this recipe we are about to see how we can modify certain aspects of our site directly from the admin panel. This way we can quickly adapt the site to our needs.

### Getting ready

If you have followed the previous chapters in this book, you might want to create a fresh Magento installation before you perform this recipe. This way it will be easier to follow as the other changes won't mess with the ones in this recipe.

### How to do it...

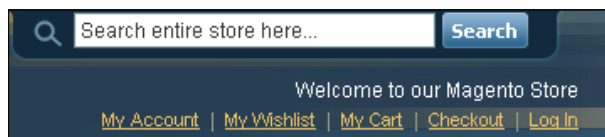
As in this recipe we will make all our changes in the admin panel. The first thing we will need to do is log into our admin panel, so let's do that and follow these steps:

1. The first thing we are going to change is the welcome message. This is quite easy. Just go to **System | Configuration | General | Design**.
2. Here we will need to open the **Header** tab, as can be seen in the following screenshot:

Header		
Logo Image Src	<input type="text" value="images/logo.gif"/>	[STORE VIEW]
Logo Image Alt	<input type="text" value="Magento Commerce"/>	[STORE VIEW]
Welcome Text	<input type="text" value="Default welcome msg!"/>	[STORE VIEW]

3. Here we can change the welcome message to just about anything we want, for example **Welcome to our Magento Store**. Change it and click on the **Save Config** button.

4. If we now refresh our frontend, we will see something like the following screenshot:



5. Note that from the same admin screen we can change the logo of our site. This is pretty easy. Note that in the **Header** tab we can see an input field with the label of **Logo Image Src**. Here we can write the path to the image.
6. As seen in the preceding screenshot it says our logo image is placed in `images/logo.gif`. Let's go to the folder where our Magento is installed and then to the `skin/frontend/default/default2` folder.
7. Inside that folder open the `images` folder. Inside you will be able to find an image called `logo.gif`. Duplicate and rename it `new_logo.gif`.
8. Modify this image in any way you want and return to the admin screen. In the **Logo Image Src** field write **`images/new_logo.gif`** and save the modifications. This page should be look the following screenshot:

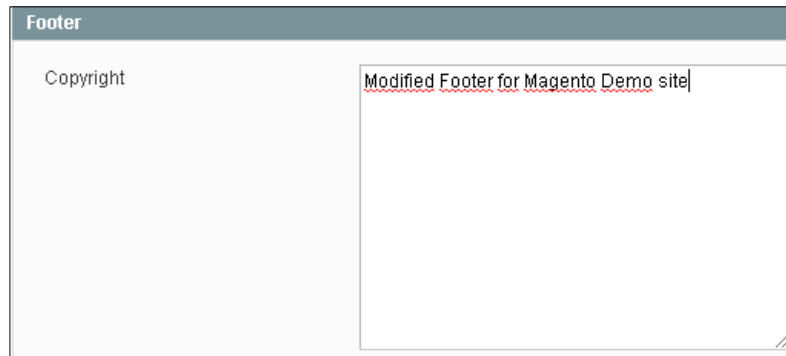
Header		
Logo Image Src	<input type="text" value="images/new_logo.gif"/>	[STORE VIEW]
Logo Image Alt	<input type="text" value="Demo Magento Store"/>	[STORE VIEW]
Welcome Text	<input type="text" value="Welcome to our Magento Store"/>	[STORE VIEW]

9. Note that we have also modified the **Logo Image Alt** field. And after saving, if we refresh our frontend we will be able to see our changes, as shown in the following screenshot:

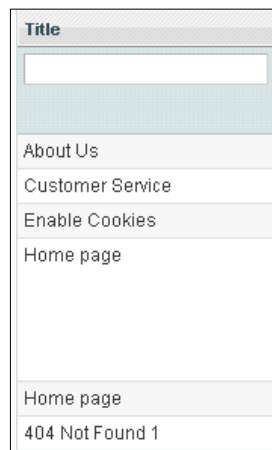




- Now that we have our logo modified to suit our needs, we can do the same with the footer. Just go to **System | Configuration | General | Design**. This time we are going to work in the **Footer** section, as can be seen in the following screenshot:



- Here we have modified the previous text with one of our liking and, after saving it we can refresh our frontend and see the changes take effect.
- Our next step is to modify the **About Us** or **Customer Service** pages and their contents. This is pretty easy. In our admin panel we need to go to the **CMS | Pages** menu.
- There we can see all the CMS pages available. More or less you will see something like the following screenshot:



- Just clicking on any of these items will open a page that will let us modify its contents. It's that easy.
- Good, let's continue. Another necessary modification is to the contact form and e-mail. Let's do that. In our admin panel go to **System | Configuration | General | Contacts**.

16. There you will be able to see a panel just like the following screenshot:

The screenshot shows a configuration panel with two main sections. The first section, titled 'Contact Us', contains a single setting: 'Enable Contact Us' with a dropdown menu currently set to 'Yes'. The second section, titled 'Email Options', contains three settings: 'Send Emails To' with a text input field containing 'hello@example.com', 'Email Sender' with a dropdown menu set to 'Custom Email 2', and 'Email Template' with a dropdown menu set to 'Contact Form (Default Template from Locale)'.

17. Here in the **Enable Contact Us** we can enable or disable the contact form. If we have enabled it, we could define the e-mail to use in the **Send Emails To** input column, the sender, and the template. Remember to save the changes after doing this.

18. But in which other places can we configure our e-mail accounts? Easy, this can be done in **System | Configuration | General | Store Email Addresses**.

19. There we can configure the following:

- ☐ The general contact e-mail
- ☐ The sales representative e-mail
- ☐ The customer support e-mail
- ☐ The custom e-mail 1
- ☐ The custom e-mail 2

20. At this point we have modified a lot of configurations of our site, all of them through our admin panel.

## How it works

In this recipe we have seen how to configure our site through the admin panel. Most of these configurations are very important and need to be done before our site is set to a production environment.

## See also

- ▶ *Step 1—laying the foundation*
- ▶ *Step 3—small modifications that can be done in layout files*

## Step 3—small modifications that can be done in layout files

In this recipe we are about to see how to modify certain elements of our theme through editing of our `layout.xml` files. Although, we can achieve a great level of customization thanks to the admin panel, it's impossible to do everything. When we can't do something through the admin panel, it's time to edit some files!

### Getting ready

If you have followed the previous chapters in this book, maybe you want to create a fresh Magento installation before this one. This way it will be easier to follow as the other changes won't mess with the ones in this recipe.

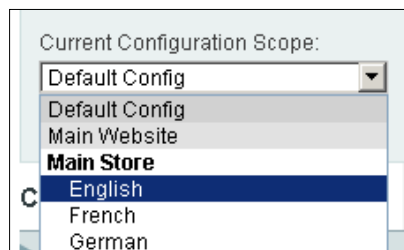
### How to do it...

This time we are going to see some coding. Let's start:

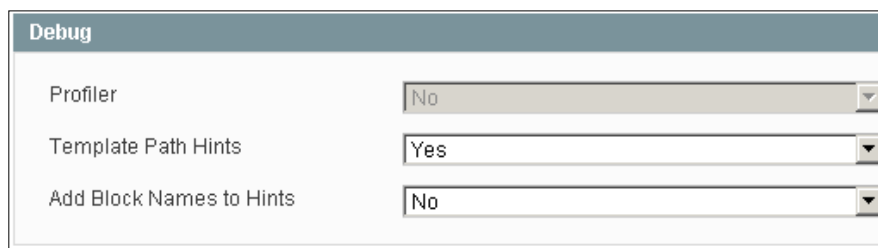
1. For example, say we want to remove the following image from our site:



2. How could we remove this image? Well, in order to help us a bit, first we will go to our admin panel. There we will go to **System | Configuration**, and select the **Configuration Scope** to the **English Store**. For example, as shown in the following screenshot:



3. After doing so go to the **Admin** tab, **Developer** option, and then we can take a look to the **Debug** panel, as can be seen in the following screenshot:



Debug	
Profiler	No
Template Path Hints	Yes
Add Block Names to Hints	No

4. Here we can select any of these options, but for now we are going to use **Template Path Hints**. This will tell us where our template elements came from. Save this configuration and refresh your website's frontend.
5. In the frontend we can see that the template path hint for the **Back to school image** is `frontend\base\default\template\callouts\right_col.phtml`.
6. We could edit this file, but as its best not to edit the files that are in the base folder we are going to duplicate it.
7. Open your Magento installation folder and go to `app/design/frontend/base/default/template/callouts`, and once there copy the file `right_col.phtml` and paste it in `app/design/frontend/default/default2/template/callouts`.
8. We can now open that file. Inside we will find something like the next piece of code:

```
<div class="block block-banner">
    <div class="block-content">
        <?php if(strtolower(substr($this->getLinkUrl(),0,4))=='http'): ?>
            <a href="<?php echo $this->getLinkUrl() ?>"
title="<?php echo $this->__($this->getImgAlt()) ?>">
                <?php elseif($this->getLinkUrl()): ?>
                    <a href="<?php echo $this->getUrl($this->getLinkUrl())
?>" title="<?php echo $this->__($this->getImgAlt()) ?>">
                        <?php endif; ?>
                            getLinkUrl()): ?> title="<?php
echo $this->__($this->getImgAlt()) ?>"<?php endif; ?> alt="<?php
echo $this->__($this->getImgAlt()) ?>" />
                                <?php if($this->getLinkUrl()): ?>
                                    </a>
                                <?php endif ?>
                            </div>
                        </div>
```

9. Removing that piece of code will effectively remove the banner, but that's not the way to do it.
10. A better way to do this is editing the layout files. In this case we would need to edit `app/design/frontend/base/default/layout/catalog.xml`. So copy this file and paste it in `app/design/frontend/default/default2/layout/catalog.xml`.
11. Open the file in the editor, and look for the following tags:

```
<default>
.
.
.
</default>
```

12. There we will find the following block:

```
<block type="core/template" name="right.permanent.
callout" template="callouts/right_col.phtml">
    <action method="setImgSrc"><src>images/media/col_
right_callout.jpg</src></action>
    <action method="setImgAlt" translate="alt"
module="catalog"><alt>Keep your eyes open for our special Back to
School items and save A LOT!</alt></action>
</block>
```

13. Again, removing or commenting this piece of code will make the callout disappear.
14. But there's another way to achieve the same thing. Create a file called `local.xml` inside `app/design/frontend/default/default2/layout`.
15. Inside that file write the following code:

```
<?xml version="1.0" ?>
<layout>
    <default>

        </default>
</layout>
```

16. If we want to remove that banner we can break it inside that block, inside the default tags by writing the following code:
17. This would also effectively remove the banner. How have we done that? In fact it is quite easy. Remember this block in `catalog.xml` file?

```
<block type="core/template" name="right.permanent.callout"
template="callouts/right_col.phtml">
    <action method="setImgSrc"><src>images/media/col_
right_callout.jpg</src></action>
```

```

        <action method="setImgAlt" translate="alt"
module="catalog"><alt>Keep your eyes open for our special Back to
School items and save A LOT!</alt></action>
    </block>

```

18. There in bold we have the block name, which can later be used in the remove tag.

19. But, what if we want to add another banner? Quite easy, the contents of our local.xml file would be as follows:

```

<?xml version="1.0" ?>
<layout>
    <default>
        <reference name="right">
            <block type="core/template" name="right.permanent.
callout_copy" template="callouts/right_col.phtml">
                <action method="setImgSrc"><src>images/media/col_
right_callout.jpg</src></action>
                <action method="setImgAlt" translate="alt"
module="catalog"><alt>Keep your eyes open for our special Back to
School items and save A LOT!</alt></action>
            </block>
        </reference>
    </default>
</layout>

```

20. This would result in the following screenshot:



21. But, this banner is not appearing near the other. That can be fixed with the following changes:

```
<?xml version="1.0" ?>
<layout>
  <default>
    <reference name="right">
      <block type="core/template" name="right.permanent.
callout_copy" template="callouts/right_col.phtml" after="right.
permanent.callout">
        <action method="setImgSrc"><src>images/media/col_
right_callout.jpg</src></action>
        <action method="setImgAlt" translate="alt"
module="catalog"><alt>Keep your eyes open for our special Back to
School items and save A LOT!</alt></action>
      </block>
    </reference>
  </default>
</layout>
```

22. Add after, and the name of the block will place our block next to it. Note that we could also do this using before.

23. What if we want to remove these blocks of one page, like the product detail page? Then we would write the following code in local.xml:

```
<?xml version="1.0" ?>
<layout>
  <default>
    <reference name="right">
      <block type="core/template" name="right.permanent.
callout_copy" template="callouts/right_col.phtml" after="right.
permanent.callout">
        <action method="setImgSrc"><src>images/media/col_
right_callout.jpg</src></action>
        <action method="setImgAlt" translate="alt"
module="catalog"><alt>Keep your eyes open for our special Back to
School items and save A LOT!</alt></action>
      </block>
    </reference>
  </default>

  <catalog_product_view translate="label">
    <reference name="right">
      <remove name="right.permanent.callout_copy" />
      <remove name="right.permanent.callout" />
    </reference>
  </catalog_product_view>
</layout>
```

24. We have created a block, `catalog_product_view`, and removed the callout form there. That's all! Now you can go to your website's frontend and see the changes take effect.

## How it works

We have seen various methods to modify some elements of our theme. Editing PHTML files is not recommended for removing blocks, but could be useful in other modifications. Other options that we have are editing some layout files, or creating a `local.xml` file with our theme customizations.

Any of these methods will help us in editing our theme.

## See also

- ▶ *Step 1—laying the foundation*
- ▶ *Step 2—small modifications that can be done in admin panel*





# Index

## Symbols

`$this->__()` method 133

## A

**Add Design Change button** 38

**Add item button** 120

**Add Root Category button** 148

**Add Subcategory button** 148

**Add to cart link** 111

**admin panel modifications**

creating, steps 168-171

**app folder** 27

## B

### blocks

adding 14, 15

content block 14

structural block 14

working 15

### breadcrumbs

about 48

adding 48, 49

working 50

## C

### cache shutdown

steps 10, 11

working 11

### catalog page

about 68

styling 69-73

template, setting 68

working 74

**catalog.xml file** 16

### CMS pages

about 74, 145

creating 145, 146

linking, with themes 32-34

template, setting 74, 76

working 76, 147, 148

**contact.php** 10

**content, adding in home page**

steps 43-47

working 48

### Content tab

**Create Store button** 149

**Create Store View button** 150

### Cufón

using, for including theme font 88-90

working 90

### current store

getting 136-138

working 140

## D

**Design Change screen** 38

**Design link** 31

**Download button** 89

### downloading

Pikachoose library 124

## E

**Extension Key** 98, 115

## F

**f002.classic\_1.zip**

folders 26

## **featured products**

adding, to home page 105-110  
working 111

## **featured products block**

about 120  
building 121-126  
modifying, for sold products display 127, 129  
working 126, 129

## **features, themes**

about 155-158  
screenshots 158

## **footer links**

about 56  
HTML code, replacing with 56, 57  
working 57

## **foundation, Magento installation**

laying 166  
laying, steps 166, 167  
working 167

## **G**

**getChildHtml method 51**

**Get Extension Key button 25**

**getSkinUrl method 21**

**getUrl method 34**

## **H**

### **handles, layouts**

starting with 18, 19  
working with 19

### **header info**

modifying 30, 31

**Header tab 168**

### **home page**

content, adding in 43-47  
featured products, adding to 105-110  
working 111

**homepage.phtml file 107**

**home.php 10**

### **HTML code**

replacing, with footer links 56, 57

## **I**

**index.php 10**

## **installation, ModuleCreator extension**

about 114  
requirements 114  
steps 114-116  
working 116

## **K**

### **key**

Extension key 98  
URL key 33

## **L**

### **language selector**

about 55  
adding 55  
working 56

### **layout file modification**

creating, steps 172-177  
working 177

### **layouts**

about 16  
creating 16, 17  
handles 18  
working 18

**left\_column.php 10**

**Let's do this button 88**

### **logo**

modifying 30, 31

**logo.gif 169**

## **M**

### **Magento**

cache shutdown 10  
theme inheritance 11

### **Magento Connect**

about 24  
theme, installing 24

**Magento Connect Manager screen 25**

**Magento Easy Lightbox 97**

**Magento Go tool 163**

### **Magento installation**

admin panel modifications 168  
foundation, laying 166  
layout file modification 172

## **Magento themes**

- files 10

## **main menu**

- about 62
- building 62, 64
- working 64

## **Manage items option 121**

## **manual theme installation**

- requirements 26, 27
- steps 27

## **mini cart**

- about 58
- adding 58-61
- working 61

## **ModuleCreator extension**

- installing 114
- using 116-118
- versions 119
- working 119, 120

## **ModuleCreator extension, installing. *See installation, ModuleCreator extension***

## **multicurrency store**

- about 140
- creating 141-144
- working 144

## **N**

**Nivo banner slider** 94

## **O**

## **own theme, building**

- breadcrumbs, adding 48
- footer links 56
- foundation, laying 37
- homepage content, adding 43
- language selector, adding 55
- main menu, building 62
- mini cart, adding 58
- requirements 36, 37
- search form, enabling 50
- top links, adding 53

## **P**

## **Pikachoose library**

- about 129

- downloading 124

## **product detail page**

- about 83
- modifying 84-86
- working 86

## **product page**

- social media sharing, adding to 101-105
- working 105

## **R**

## **recently installed theme**

- selecting 27, 28
- working 28

## **S**

## **Save config button 28**

## **search form**

- about 50
- enabling 50-52
- working 52

## **Settings panel 115**

## **Shopping cart page**

- about 81
- template, setting 81, 83
- working 83

## **Show/Hide Editor button 45, 106**

## **Sign In page**

- template, setting 76-80
- working 81

## **simple extensions 113**

## **site aspects**

- enhancing 66, 67
- page.xml file, working 67

## **skin folder 27**

## **SlideDeck content slider 90**

## **T**

## **template**

- setting, for catalog page 68-74
- setting, for CMS pages 75
- setting, for Shopping cart page 81-83
- setting, for Sign In page 76-81

## **template customization**

- about 20

- steps 20
- working 21
- template path hints**
  - about 29
  - enabling 29
  - working 30
- theme foundation**
  - getChildHtml method 43
  - laying 37-43
- theme inheritance, Magento**
  - about 11
  - theme, creating 12, 13
  - theme, selecting 12, 13
  - theme, working 13
- theme installation**
  - manually 26
  - through Magento Connect 24-26
- theme installation, through Magento Connect**
  - requirements 24
  - steps 24-26
  - working 26
- themes**
  - building 35
  - Cufón, using 88
  - features 155
  - ideas 159
  - linking, with CMS pages 32-34
  - localizing 131-135
  - localizing, Magento method used 135
  - MAG themes 160
  - MOJO themes 159
  - net.tutsplus 160
  - packing 153-155
  - selling, locations 159
  - themeforest 159
  - WOO themes 160
- top links**
  - about 53
  - adding 53, 54
  - working 54
- topmenu**
  - about 148
  - translating 148-152
  - working 152

**U**

**URL Key 33**

**V**

**view.phtml file 104**

**Y**

**yoast tweaking websites 162**

**Z**

**ZIP file 153, 155**



## **Thank you for buying Magento 1.4 Theming Cookbook**

### **About Packt Publishing**

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: [www.packtpub.com](http://www.packtpub.com).

### **About Packt Open Source**

In 2010, Packt launched two new brands, Packt Open Source and Packt Enterprise, in order to continue its focus on specialization. This book is part of the Packt Open Source brand, home to books published on software built around Open Source licences, and offering information to anybody from advanced developers to budding web designers. The Open Source brand also runs Packt's Open Source Royalty Scheme, by which Packt gives a royalty to each Open Source project about whose software a book is sold.

### **Writing for Packt**

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to [author@packtpub.com](mailto:author@packtpub.com). If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.



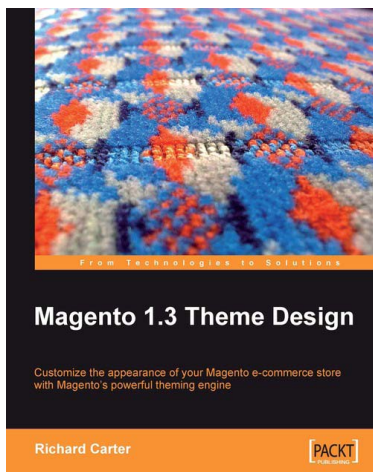
## Magento 1.4 Themes Design

ISBN: 978-1-84951-480-4

Paperback: 292 pages

Customize the appearance of your Magento 1.4 e-commerce store with Magento's powerful theming engine

1. Install and configure Magento 1.4 and learn the fundamental principles behind Magento themes
2. Customize the appearance of your Magento 1.4 e-commerce store with Magento's powerful theming engine by changing Magento templates, skin files and layout files
3. Change the basics of your Magento theme from the logo of your store to the color scheme of your theme



## Magento 1.3 Theme Design

ISBN: 978-1-847196-64-4

Paperback: 188 pages

Customize the appearance of your Magento e-commerce store with Magento's powerful theming engine

1. Give your Magento stores a unique branded look and feel by creating your own Magento themes
2. Use design techniques to reinforce your brand message and increase sales
3. Customize your Magento theme's look, feel, layout, and features
4. Promote and improve your Magento store with the use of social media such as Twitter, social bookmarks, and so on

Please check [www.PacktPub.com](http://www.PacktPub.com) for information on our titles



## Magento 1.3 Sales Tactics Cookbook

ISBN: 978-1-849510-12-7

Paperback: 292 pages

Solve real-world Magento sales problems with a collection of simple but effective recipes

1. Build a professional Magento sales web site, with the help of easy-to-follow steps and ample screenshots, to solve real-world business needs and requirements
2. Develop your web site by using your creativity and exploiting the sales techniques that suit your needs
3. Provide visitors with attractive and innovative features to make your site sell



## Magento: Beginner's Guide

ISBN: 978-1-847195-94-4

Paperback: 300 pages

Create a dynamic, fully featured, online store with the most powerful open source e-commerce software

1. Step-by-step guide to building your own online store
2. Focuses on the key features of Magento that you must know to get your store up and running
3. Customize the store's appearance to make it uniquely yours
4. Clearly illustrated with screenshots and a working example

Please check [www.PacktPub.com](http://www.PacktPub.com) for information on our titles