



Quick answers to common problems

Drupal 7 Cookbook

Over 70 recipes that will advance your Drupal skills from novice to pro

Dylan Spencer James

[PACKT] open source[®]
PUBLISHING community experience distilled

Drupal 7 Cookbook

Over 70 recipes that will advance your Drupal skills from novice to pro

Dylan James



BIRMINGHAM - MUMBAI

Drupal 7 Cookbook

Copyright © 2012 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: July 2012

Production Reference: 1050712

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

ISBN 978-1-84951-796-6

www.packtpub.com

Cover Image by Asher Wishkerman (a.wishkerman@mpic.de)

Credits

Author	Project Coordinator
Dylan James	Michelle Quadros
Reviewers	Proofreader
Jacques R. Blier	Aaron Nash
Anand Narayanaswamy	Indexer
Veturi JV Subramanyeswari	Hemangini Bairi
Acquisition Editor	Graphics
Sarah Cullington	Manu Joseph
Lead Technical Editor	Production Coordinator
Shreerang Deshpande	Nilesh R. Mohite
Technical Editor	Cover Work
Madhuri Das	Nilesh R. Mohite

About the Author

Dylan James is an open source developer from and based in South Wales. After gaining a Computer Science degree from Cardiff University, Dylan got experience as a database developer for Walkers snack foods before going on to work at the Welsh media company, Tinopolis Interactive, where he worked as a Lead Flash developer on some large projects for clients such as the BBC, LearnDirect, the MoD, and The Cambridge University Press. It was at Tinopolis that Dylan got his first taste of developing using open source platforms, delivering sites for clients using Joomla and Moodle.

After leaving Tinopolis in 2010, Dylan began working at a web agency in Swansea, creating Drupal websites for some high profile clients.

For updates and errata, please visit www.dylanspencerjames.com

If I would like to thank my parents, Janice and Julian, my sister Lauren, her husband Kristian, and all of my friends, for all of their encouragement and support, and the occasional sanity check!

About the Reviewers

Jacques R. Blier, with his background in typography and communications, got quickly interested in Internet technologies. Since 1994, he has participated in the creation and implementation of Internet websites, and also provided training on new technologies. In 2004 he created his own computer services and communications business, xMac info.

With extensive experience in sales, technical support, training, networking, and user interface design for software used by cellular operators, Jacques became interested in Drupal, since 2005. Since then, he has been using this development platform for creating and developing all types of websites. He specializes in CSS and prefers to create the Drupal themes that people ask him, from scratch.

Anand Narayanaswamy is an ASPIInsider who works as a freelance technical writer based in Trivandrum, India. He had worked as a technical editor/reviewer for various publishers such as Sams, Addison-Wesley, Mc Graw Hill, Packt, and ASPAlliance.com. Anand has expertise in the installation, management, and usage of popular ASP.NET and PHP based blogs/Content Management Systems (CMS). He is the author of *Community Server Quickly* (www.packtpub.com/community-server/book) published by Packt Publishing and runs www.learnxpress.com.

First, I would like to thank the almighty for giving me the strength and energy to work every day. I specially thank my father, mother, and brother for providing valuable help, support, and encouragement. I also thank Michelle Quadros, Project Coordinator, Packt Publishing for her assistance, cooperation, and understanding throughout the review process of this book.

Sree (also known as **Veturi JV Subramanyeswari**) is currently working as Drupal Architect at a well known software consulting MNC in India. Post joining this company she served a few Indian MNCs, many start ups, and R&D sectors in various roles such as programmer, tech lead, and research assistant. She has around 8 years of work experience in web technologies covering media and entertainment, publishing, healthcare, enterprise architecture, manufacturing, public sector, defense communication, gaming, and so on. She is also a well known speaker who delivers talks on Drupal, Open Source, PHP, Women in Technology, and so on.

She reviewed other tech books such as the following :

- ▶ Drupal 7 Multi Sites Configuration
- ▶ Building Powerful and Robust Websites with Drupal 6
- ▶ Drupal 6 Module development
- ▶ PHP Team Development
- ▶ Drupal-6-site-blueprints
- ▶ Drupal 6 Attachment Views
- ▶ Drupal E-Commerce with Ubercart 2.x
- ▶ Drupal 7: First Look

I would like to thank my family and friends, who supported me in completing my reviews on time to a high quality.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit 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 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 for more details.

At 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, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface	1
Chapter 1: Installing and Configuring Drupal	7
Introduction	7
Installing Drupal	8
Installing Drupal distributions	13
Installing modules and themes	14
Setting up site search	16
Creating a multi-site Drupal installation	18
Chapter 2: Creating and Publishing Content	21
Introduction	21
Creating a basic page and adding it to the main menu	22
Installing and configuring a WYSIWYG editor	26
Editing existing content and adding an image	29
Adjusting the tabbed settings for content nodes	33
Configuring comments	36
Publishing an RSS feed of basic pages	39
Chapter 3: Working with Blocks	41
Introduction	41
Adding a new block	42
Creating a submenu block	45
Creating a Superfish menu block	49
Creating a block with Views	52
Adding a new block region to a theme	55
Creating a mega-footer menu	59
Conditional display of a block	64

Chapter 4: Custom Content Types	67
Introduction	67
Creating a basic content type	68
Configuring the output of a content type	71
Applying an image format	73
Installing more field types	75
Creating a more advanced content type	79
Building a custom content importer	85
Building a forum	89
Chapter 5: Using Views to Create Custom Lists, Grids, and Tables	93
Introduction	93
Creating a news listing view	94
Creating a dynamic view	98
Creating a latest news block	104
Creating a news image grid view	107
Creating a randomly selected list of images	111
Creating an archived content block and view	113
Building complex views using relationships	118
Adding a text search filter to a view	122
Using attachments to extend Views' output	124
Chapter 6: Creating Flexible Pages Using Panels	129
Introduction	129
Adding custom text to a page	130
Adding a block to a page	134
Adding a dynamic view to a page	137
Configuring the visibility of the page	140
Creating a custom page layout using the Layout builder	143
Chapter 7: Working with Media	149
Introduction	149
Creating a simple slideshow carousel	150
Creating a document content type	155
Creating a simple document library	158
Linking documents to a content type	161
Adding video to a content type	165
Chapter 8: Integrating Web APIs	171
Introduction	171
Integrating with Facebook	172
Displaying a live Twitter feed	176
Adding simple PayPal integration to content types	181

Table of Contents

Setting up the Add this social bookmarking service	187
Adding a Google Map to content	190
Chapter 9: Creating Regular, Mobile, and Tablet Themes	197
Introduction	197
Creating a new theme using Zen	198
Overriding HTML output of a content type	202
Creating a "bare-bones" theme from scratch	211
Using the Mobile tools module	215
Installing an off-the-shelf mobile and tablet theme	218
Configuring theme compression and caching	221
Chapter 10: Working with Other Languages	225
Introduction	225
Installing another language using Locale	226
Managing interface translation using Locale	230
Enabling content type translation	232
Displaying a language switching block for end users	235
Creating a multilingual View	237
Chapter 11: Managing Users	243
Introduction	243
Creating new user accounts	244
Managing user roles	246
Setting up a new user notification	249
Adding a biography field to the user profile	253
Building a grid view of profile pictures	258
Chapter 12: Running Drupal	269
Introduction	269
System maintenance	270
Setting up a backup system	274
Search Engine Optimization (SEO) with Drupal	284
Securing a Drupal installation	288
Configuring Drupal caching	290
Running commands with the Drush tool	294
Index	301

Preface

Drupal 7 is a modern Content Management System, famous for its flexibility and power. Using Drupal you can easily create custom functionality that would otherwise have to be purchased in many of the other leading CMSs.

Drupal 7 Cookbook is filled with recipes to help you to do more with Drupal and improve your skills. Chapters range from content creation, to theming, to managing your site. You will learn how to create your own content types and use them to create Views, Blocks, and Pages. This book will take you from novice to pro in just 12 chapters.

In a wide variety of practical recipes, you will learn how to work with views and panels, how to provide translations for your content to create a multilingual site, and to integrate your site with social media. You can develop the Zen starter theme or learn how to create custom cross-browser compatible Drupal themes, including themes for mobile devices. The *Drupal 7 Cookbook* contains all of the means necessary to take your skills from those of a novice Drupal user to a proficient site builder.

What this book covers

Chapter 1, Installing and Configuring Drupal, begins with the Drupal fundamentals seeing how to install and configure a standard Drupal installation.

Chapter 2, Creating and Publishing Content, introduces the principles and methods of creating content, in addition to exploring the attributes and features common to Drupal content nodes, such as comments.

Chapter 3, Working with Blocks, explains how to create a range of blocks, from the basic content block, to the more advanced blocks used in menus and blocks created from views.

Chapter 4, Custom Content Types, explains how to create and configure simple and advanced custom content types.

Chapter 5, Using Views to Create Custom Lists, Grids, and Tables, explains how to create a wide range of views for uses such as news listings and image galleries.

Chapter 6, Creating Flexible Pages using Panels, explores the various ways to make use of panels. We see how to add content to panels, and how we can customize their layout for any situation.

Chapter 7, Working with Media, explains how to manipulate and utilize media files within your custom content types.

Chapter 8, Integrating Web APIs, explains how we can integrate with many of the social sites, such as Facebook and Twitter, and also how to integrate with Google maps, and the Add This bookmarking service.

Chapter 9, Creating Regular, Mobile, and Tablet Themes, explains how to build a new theme using a base theme, and then how to optimize and customize a theme.

Chapter 10, Working with Other Languages, explains how to use the third-party language modules that provide localization for all content within the system. Additionally, we explore how to manage languages, and how to switch between languages.

Chapter 11, Managing Users, explores the main tasks involved with user management, from creating and editing user accounts, to managing roles and permissions.

Chapter 12, Running Drupal, explores the issues relating to the day-to-day administration, including, updating modules, security, and backup.

What you need for this book

To run the examples in the book following software will be required:

Web Server:

Apache recommended (The Drupal installation recipe is specific to an Apache installation).

- ▶ Apache 1.3 or Apache 2.x, Apache 2.x recommended
- ▶ Microsoft IIS 7

#	Software Name	URL
1	Apache	http://httpd.apache.org/
2	Microsoft IIS	http://www.iis.net/

Database:

- ▶ MySQL 5.0.15 or higher with PDO, SQLite 3.3.7 or higher

Microsoft SQL Server and Oracle are supported by an additional module

PHP:

- ▶ PHP 5.2.5 or higher, PHP 5.3 recommended

#	Software Name	URL
1	PHP	http://php.net

FTP client:

There are a number of free FTP clients; this is the recommended, cross-platform client:

#	Software Name	URL
1	FileZilla	http://filezilla-project.org/

Text editor:

For editing local PHP, HTML, and CSS files, it's useful, but not essential, to use a tool such as Dreamweaver or Notepad++. Of course, you can always use a native text editor such as Notepad, if you would prefer.

#	Software Name	URL
1	Dreamweaver	http://www.adobe.com/products/dreamweaver.html (commercial)
2	Notepad++	http://notepad-plus-plus.org/
3	Eclipse	http://www.eclipse.org/

Who this book is for

This book is for people who are familiar with the concepts of web development and CMS. If you have previous experience of working with Drupal, then this book will further your knowledge and give you ideas for fun new things to do with Drupal.

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: "Ensure that register globals is set to off in the `PHP.ini` file."

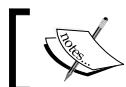
A block of code is set as follows:

```
name = Bartik extra
base theme=bartik
description=A Bartik sub-theme which provides a navigation region
core = 7.x
```

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

```
<?php
    // We hide the comments and links now so that we can render them
    later.
    hide($content['comments']);
    hide($content['links']);
?><div class="recipe-instructions"><?php print
render($content['body']);?></div><?php
?><div class="recipe-ingredients"><?php print
render($content['field_recipe_ingredients']);?></div><?php
print render($content);
?>
```

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 **Standard** option and then select **Save and continue**."



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, and mention the book title through the subject of your message.

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.

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 to our website, or added to any list of existing errata, under the Errata section of that title.

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 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 if you are having a problem with any aspect of the book, and we will do our best to address it.

1

Installing and Configuring Drupal

In this chapter, we will cover:

- ▶ Installing Drupal
- ▶ Installing Drupal distributions
- ▶ Installing modules and themes
- ▶ Setting up the site search
- ▶ Creating a multi-site Drupal installation

Introduction

Throughout this chapter, we will explore the process of setting up a Drupal website. We will see how to install the official Drupal 7 release and also how to install third-party Drupal distributions which are preconfigured in a variety of ways.

Following the installation, we will move on to the installation of third-party modules and then how your site can be configured to provide a site search form.

The chapter culminates with a recipe on setting up Drupal multi-site where two Drupal sites are configured to run from the same core files.

Installing Drupal

There are a number of different ways to install Drupal on a web server, but in this recipe we will focus on the standard, most common installation, which is to say, Drupal running on an Apache server, which runs PHP with a MySQL database. In order to do this we will download the latest Drupal release, and walk you through all of the steps required to get it up and running.

Getting ready

Before beginning, you need to ensure that you meet the following minimal requirements:

- ▶ Web hosting with FTP access (or file access through a control panel).
- ▶ A server running PHP 5.2.5+ (5.3+ recommended).
- ▶ A blank MySQL database and the login credentials to access it.
- ▶ Ensure that *register globals* is set to off in the PHP .ini file. You may need to contact your hosting provider to do this.

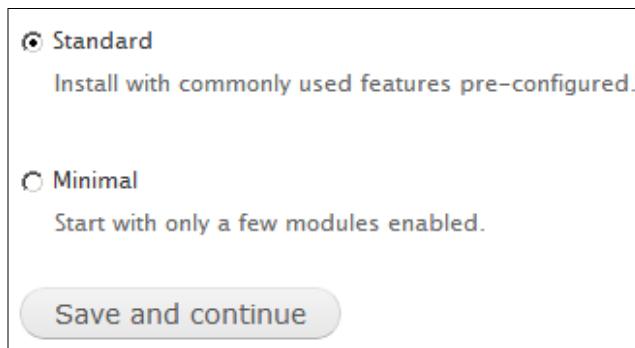
How to do it...

1. The first step is to download the latest Drupal 7 release from the Drupal download page, which is located at <http://drupal.org/project/drupal>:

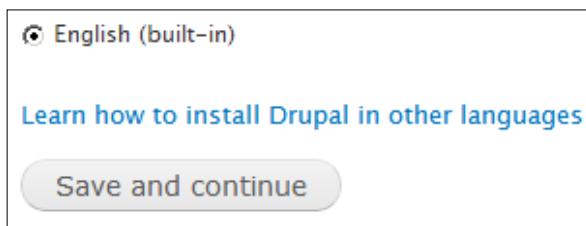
Downloads				
Recommended releases				
Version	Downloads	Date	Links	
7.14	tar.gz (2.98 MB) zip (3.4 MB)	2012-May-02	Notes	
6.26	tar.gz (1.05 MB) zip (1.22 MB)	2012-May-02	Notes	
Development releases				
Version	Downloads	Date	Links	
7.x-dev	tar.gz (2.98 MB) zip (3.41 MB)	2012-May-09	Notes	
6.x-dev	tar.gz (1.05 MB) zip (1.22 MB)	2012-May-03	Notes	

This page displays the most recent and *recommended* releases for both Drupal 6 and 7. It also displays the most recent *development* versions, but be sure to download the recommended release (development versions are for developers who want to stay on the cutting edge).

2. When the file is downloaded, extract it and upload the files to your chosen web server document root directory on the server. This may take some time.
3. Configure your web server document root and server name (usually through a vhost directive).
4. When the upload is complete, open your browser and in the address bar, type in the server name configured in the previous step to begin the installation wizard.
5. Select **Standard** option and then select **Save and continue**:



6. The next screen that you will see is the language selection screen; there should only be one language available at this point. Ensure that **English** is selected before proceeding:



7. Following a requirements check, you will arrive at the database settings page. Enter your database name, username, and password in the required fields. Unless your database details have been supplied with a specific host name and port, you should leave the advanced options as they are and continue.
8. You will now see the **Site configuration** page. Under **Site information** enter the name you would like to appear as the site's name.
9. For **Site e-mail address** enter an e-mail address.

10. Under the **SITE MAINTENANCE ACCOUNT** box, enter a username for the admin user (also known as user 1), followed by an e-mail address and password:

The screenshot shows the Drupal installation configuration interface. On the left, there's a sidebar with a blue water drop logo and a list of completed steps: Choose profile, Choose language, Verify requirements, Set up database, Install profile, and Configure site (which is currently selected). Below that is a 'Finished' link. The main area is divided into two sections: 'SITE INFORMATION' and 'SITE MAINTENANCE ACCOUNT'. In the 'SITE INFORMATION' section, the 'Site name' is set to 'My Drupal Site' and the 'Site e-mail address' is 'info@example.com'. A note explains that automated emails will be sent from this address. In the 'SITE MAINTENANCE ACCOUNT' section, the 'Username' is 'admin', and the 'E-mail address' is 'admin@example.com'. The 'Password' field contains '.....' and is marked as 'Fair' strength. The 'Confirm password' field also contains '.....' and is marked as 'Passwords match: yes'. At the bottom, there's a note about making the password stronger with options like uppercase letters, numbers, and punctuation.

SITE INFORMATION

Site name *
My Drupal Site

Site e-mail address *
info@example.com

Automated e-mails, such as registration information, will be sent from this address. Use an address ending in your site's domain to help prevent these e-mails from being flagged as spam.

SITE MAINTENANCE ACCOUNT

Username *
admin

Spaces are allowed; punctuation is not allowed except for periods, hyphens, and underscores.

E-mail address *
admin@example.com

Password *
..... Password strength: Fair

Confirm password *
..... Passwords match: yes

To make your password stronger:
• Add uppercase letters
• Add numbers
• Add punctuation

11. In the **Server settings** box, select your country from the drop-down, followed by your local time zone.
12. Finally, in the **Update notification** box, ensure that both options are selected. Click on **Save and continue** to complete the installation. You will be presented with the congratulations page with a link to your new site.

How it works...

On the server requirements page, Drupal will carry out a number of tests. It is a requirement that PHP "register globals" is set to off or disabled.

Register globals is a feature of PHP which allows global variables to be set from the contents of the Environment, GET, POST, Cookie, and Server variables. It can be a major security risk, as it enables potential hackers to overwrite important variables and gain unauthorized access.

The **Configure site** page is where you specify the site name and e-mail addresses for the site and the admin user. The admin e-mail address will be used to contact the administrator with notifications from the site, and the site e-mail address is used as the originating e-mail address when the site sends e-mails to users. You can change these settings later on in the **Site information** page in the **Configuration** section.

It's important to select the options to receive the site notifications so that you are aware when software updates are available for your site core and contrib modules; important security updates are available from time to time.

There's more...

In this recipe we have seen a regular Drupal installation procedure. There are various different ways to install and configure Drupal. We will explore some of these alternatives in the following sections. We will also cover some of the potential pitfalls you may come across with the requirements page.

Uploading through a control panel

If your web-hosting provider provides web access to your files through a control panel such as CPanel, you can save time by uploading the compressed Drupal installation package and running the unzip function on the file, if that functionality is provided. This will dramatically reduce the amount of time taken to perform the installation.

Auto-installers

There are other ways in which Drupal can be installed. Your hosting may come with an auto-installer such as Fantastico De Luxe or Softaculous. Both of these services provide a simple way to achieve the same results without the need to use FTP or to configure a database.

Database table prefixes

At the database setup screen there is an option to use a table prefix. Any prefix entered into the field would be added to the start of all table names in the database. This means that you could run multiple installations of Drupal, or possibly other CMSs from the same database by setting a different prefix. This method, however, will have implications for performance and maintenance.

Installing on a Windows environment

This recipe deals with installing Drupal on a Linux server. However, Drupal runs perfectly well on an IIS (Windows) server. Using Microsoft's WebMatrix software, it's easy to set up a Drupal site.

<http://www.microsoft.com/web/drupal>

Alternative languages

Drupal supports many different languages. You can view and download the language packs at <http://localize.drupal.org/download>.

You then need to upload the file to Drupal root/profiles/standard/translations. You will then see the option for that new language in the language selection page of the installation.

Verifying the requirements page

If all goes to plan, and the server is already configured correctly, then step 3, the server requirements page, will be skipped. However, you may come across problems in a few areas:

- ▶ Register Globals: This should be set to off in the `php.ini` file. This is very important in securing your site. If you find that register globals is turned on, then you will need to consult your hosting provider's documentation on this feature in order to switch it off.
- ▶ Drupal will attempt to create the following folder: `Drupal_root/sites/default/files`. If it fails, you may have to manually create this file on the server and give it the permission 755.
- ▶ Drupal will attempt to create a `settings.php` file by copying the `default.settings.php` file.
- ▶ If Drupal has trouble doing this, copy the `default.settings.php` file in the following directory: `Drupal_root/sites/default/default.settings.php` and rename the copied file as `settings.php`.
- ▶ Give `settings.php` full write access CHMODD 777. After Drupal finishes the installation process, it will try to set the permission of this file to 444; you must check that this has been done, and manually set the file to 444, if it has not.

See also

See *Installing Drupal distributions* for more installation options using a preconfigured Drupal distribution.

For more information about installing Drupal, see the installation guide at [Drupal.org](http://drupal.org):

<http://drupal.org/documentation/install>

Installing Drupal distributions

Drupal distributions are third-party Drupal packages where the official Drupal release has been re-packaged to contain extra features. Drupal distributions can configure a complete Drupal site ready for a specific task, eliminating the need to research and install third-party modules.

Getting ready

The requirements for installing Drupal distributions are the same as for installing the Drupal core, as described in the preceding recipe *Installing Drupal*. You will need the following:

- ▶ Web-hosting with FTP access (or file access through a control panel)
- ▶ A server running PHP 5.2.5+ (5.3+ recommended)
- ▶ An empty MySQL database and the login credentials to access it

How to do it...

1. For this example we will use the Commerce Kickstart distribution. Download this installation package from here: http://drupal.org/project/commerce_kickstart
2. Configure your web server's server name and document root.
3. After the download is complete, extract the files, and upload the files to your web space document root using your FTP client of choice.
4. Once the upload is complete, navigate to the root (using the server name configured in your web server) of your website, using your browser to start the installation wizard.
5. Complete all the steps up to the **Site configuration** (see the previous recipe, *Installing Drupal*).
6. After completing the site configuration step you will arrive at a screen entitled **Example store** that provides installation options specific to the Drupal Commerce Kickstart distribution. Select both of the boxes on this screen to generate sample content for the Drupal commerce site.

How it works...

Now that you've learned how to install Drupal distributions, the distribution world is your oyster! There is an increasing range of different Drupal distributions to choose from that cater for a wide variety of websites, from conference organization to e-learning.

See also

- ▶ *Installing Drupal*
- ▶ There are a number of different distributions currently available. Visit the following page to explore the distributions listed on the official Drupal.org site: <http://www.drupal.org/project/installation+profiles>
- ▶ <http://drupaldistrowatch.com/>

Installing modules and themes

It is entirely possible to run a very successful Drupal website out of the box without adding any third-party modules. However, this approach is limited, and your needs may grow out of the built-in core functionalities. With just a few additional modules, the functionality of your site can be significantly expanded.

In this recipe, we will first see how to install the Link module from the [Drupal.org](http://drupal.org) website, followed by how to activate and configure it. The Link module adds a **Link** field to the list of available fields that can be used to create new content types.

How to do it...

1. Go to <http://drupal.org/project/link>.
2. Examine the available releases of Link, and find the most recent Drupal 7 release; copy the URL for the `.tar.gz` file.
3. In Drupal, select **Modules** from the admin menu, followed by **Install new module**.
4. Paste the URL into the **Install from a URL** field and click on **Install**.
5. After the module is installed, select **Enable newly added modules**.
6. Look through the list of installed modules for the **FIELDS** fieldset. Select the checkbox for **Link**.

7. Click on **Save configuration** to finish:

FIELDS				
ENABLED	NAME	VERSION	DESCRIPTION	OPERATIONS
<input checked="" type="checkbox"/>	Link	7.x-1.0	Defines simple link field types.	

MEDIA				
ENABLED	NAME	VERSION	DESCRIPTION	OPERATIONS
<input checked="" type="checkbox"/>	IMCE	7.x-1.5	An image/file uploader and browser supporting personal directories and user quota. Required by: IMCE Wysiwyg API bridge (enabled)	Permissions Configure

USER INTERFACE				
ENABLED	NAME	VERSION	DESCRIPTION	OPERATIONS
<input checked="" type="checkbox"/>	IMCE Wysiwyg API bridge	7.x-1.0	Makes IMCE available as plugin for client-side editors integrated via Wysiwyg API. Requires: IMCE (enabled), Wysiwyg (enabled)	
<input checked="" type="checkbox"/>	Wysiwyg	7.x-2.1	Allows to edit content with client-side editors. Required by: IMCE Wysiwyg API bridge (enabled)	Configure

[Save configuration](#)

How it works...

We begin by going to the project page for the module we are installing. In this case, the Link module. We then look for the most recent recommended release of the module.



On the module's page it is likely that there will be two releases for both the Drupal 6 and Drupal 7 versions of the module. The files with the green background are the recommended releases. Be sure to use this version wherever possible. Sometimes there will only be an alpha or development release; try to get the highest version number.

After copying the link to the module's file, we go to the module installation page, and paste the URL to the module's file, and click on Install. Drupal then copies the file to the server, and extracts it into its own directory in the `modules` folder of the site.

After the module has been copied, we go back to the modules list, and enable the newly installed module by checking its checkbox and clicking on **Save configuration** to finish.

There's more...

There are more ways to install modules in Drupal. It's up to you to decide which method you prefer, but some other methods are described in the following sections.

Manually installing a module

The preceding method described uses the install from URL method. You may want to install a module manually. In which case simply download and extract the module package and move the extracted files to the Drupal root/sites/all/modules directory.

Installing a module with the Drush command-line tool

The process of installing modules can be made even easier by using the Drush tool to issue installation commands.

See also

- ▶ *Running commands with the Drush tool* recipe in Chapter 12, *Running Drupal*

Setting up site search

In this recipe, we will see how to configure Drupal's native site search. The search feature is a part of the Drupal core module set. We will start by enabling the module, and then proceed to set up the search index. Finally, we will move the search block to the header.

How to do it...

1. Log in to your site and select **Modules** from the admin menu.
2. Locate the **Search** module, which is under the **Core** category, and make sure the checkbox is selected.
3. To enable the module, scroll to the bottom of the page and click on **Save configuration**.
4. Select **Configuration** from the admin menu.
5. Select **Search settings** from the **SEARCH AND METADATA** category.
6. Under **INDEXING THROTTLE** leave the **Number of items to index per cron run** at **100**.
7. Under the **INDEXING SETTINGS** leave the **Minimum word length to index** as **3** and leave **Simple CJK handling** checked.
8. Under **Active search modules** ensure that **Node and User** are checked.
9. Under **Default search module** ensure that **node** is selected.
10. Under **CONTENT RANKING** set **Keyword relevance** to **1**.

11. Click on **Save configuration** and close the admin panel when the save is complete.
12. To place the **Search form** in the header, select **Structure** from the admin menu.
13. Select **Blocks** from the **Structure** menu to go to the blocks management page.
14. Search through the blocks list to find the **Search form** block and set its position to **header** by updating the drop-down menu.
15. Scroll down the page and click on **Save blocks**.

The screenshot shows two sections of the 'Search configuration' page:

- INDEXING THROTTLE**: A section with a dropdown menu set to 100. Below it is a note: "The maximum number of items indexed in each pass of a [cron maintenance task](#). If necessary, reduce the number of items to prevent timeouts and memory errors while indexing."
- INDEXING SETTINGS**: A section containing a note: "Changing the settings below will cause the site index to be rebuilt. The search index is not cleared but systematically updated to reflect the new settings. Searching will continue to work but new content won't be indexed until all existing content has been re-indexed." Below this is another note: "The default settings should be appropriate for the majority of sites." It includes a "Minimum word length to index" input set to 3, a checked checkbox for "Simple CJK handling", and a descriptive note about applying a simple tokenizer for Chinese/Japanese/Korean languages.

How it works...

The site search works by periodically indexing the pages of a site. The indexing is activated by a cron job at a predetermined interval.

Cron is the process that periodically activates tasks such as search indexing, checking for updates, and clearing the cache. Drupal 7 has its own method to periodically call the cron script; however, it's more preferable to call Drupal's cron script directly from the server by setting up a crontab for optimum performance.

It is important that you ensure that the cron is configured to run at a suitable interval. If you have lots of frequently changing content, then you will want to configure the cron to run more frequently, or less frequently if your content changes rarely. It's a trade-off between server load and search results freshness.

The **Indexing status** category on the search configuration page displays the percentage of content indexed. This indicates how much of the eligible content is indexed for the search function. Don't worry if your content isn't immediately indexed. The indexer is configured to only index a finite number of pages per cron run. The number of pages indexed per run is set in the **Indexing throttle** category. You must be careful that the throttle is not set so high that it can overload your server.

You can manually activate the cron job, and therefore, activate the search indexing procedure, by going to **Configuration | Cron | Run cron**.

The **Content ranking** option will determine the prominence of each of the listed factors; a weight of 0 being no prominence and a weight of 10 being highest prominence. The actual settings that you choose for this should depend on the type of search results you or your organization favor.

The search form block will be set to display in the first column by default. Moving it to the header is just a personal preference, with the added purpose of demonstrating how easy it is to move the block.

Creating a multi-site Drupal installation

Drupal's multi-site feature allows multiple sites to run from the same core code, while having separate databases, configurations, and user files. Multi-site functionality removes the pain in managing multiple cores and multiple sets of modules, meaning that there is only one primary installation that needs to be updated and managed. It also reduces the amount of disk space that is required.

In this recipe, we will see how to prepare the secondary site's domain so that it points to the primary site's Drupal installation. We will then see how to activate and install the multi-site functionality on the primary site.

Getting ready

You will need the following:

- ▶ An existing Drupal installation for your primary site
- ▶ A domain name for your secondary site
- ▶ The ability to update your secondary domain to point to the name server of your primary domain

- ▶ You will need the ability to create an add-on domain for your primary site (we will see how to actually create the add-on domain in the recipe)
- ▶ A blank database

How to do it...

1. Update your secondary domain's name server to be the same as the primary domain's name server.
2. On your primary server, log in to your server's admin panel and create a new add-on domain. Add the URL of the secondary domain as the new domain name.
3. Add a username and password. This is to enable you to provide specific FTP access to the secondary domain without providing FTP access for your primary domain. Set the document root to be the location on the server of your primary Drupal installation:

New Domain Name: ✓

Subdomain/FTP Username: ✓

Document Root: ✓

Password: ! Password strength must be at least 1.

Password (Again): ✓

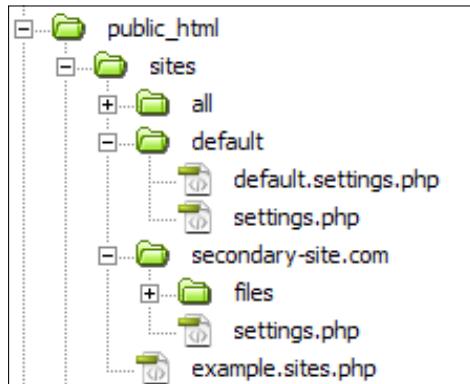
Strength (why?): Very Weak (2/100) >Password Generator

Add Domain

Hint: This feature must be enabled for your account before you can use it. Addon domains will not function unless the domain name is registered with a valid registrar and configured to point to the correct DNS servers.

4. On your primary site create a new folder inside your sites directory which has the same name as your secondary site, for example, /Drupal_root/sites/secondary-site.com/.
5. Copy default.settings.php from the following folder: /Drupal_root/default/ to the /Drupal_root/sites/secondary-site.com/ folder.

6. Rename the newly copied file to `settings.php` and set its permission to 666:



7. Under `/Drupal root/sites/secondary-site.com/`, create a new folder called `files` and set the folder permission to 777.
8. Go to your secondary site in your browser; you will see the standard installation screen.
9. Complete the installation wizard as described in the first recipe of this chapter (*Installing Drupal*). On the database settings page enter the login credentials of your newly created empty database.
10. After completing the installation, set the permissions of `Drupal root/sites/secondary-site.com/settings.php` to 444.

How it works...

Updating the name server is usually done through the domain registrar. The name server you need to enter for the secondary domain will most likely be provided by your hosting provider for your primary domain in the welcome e-mail. You may need to wait some time for the DNS records to propagate before your domain name resolves to the newly appointed server.

Creating an add-on domain allows you to set up a third-party domain to point to your hosting directory without having to redirect the user. This means that the URL in the address bar of the user's browser will remain constant, even though they are accessing the directory of another site. The setup procedure for configuring an add-on domain may be different with different hosting providers and control panels, and this example references the cPanel management panel. If you find difficulty getting this to work, reference your hosting provider's documentation for add-on domains.

2

Creating and Publishing Content

In this chapter we will cover:

- ▶ Creating a basic page and adding it to the main menu
- ▶ Installing and configuring a WYSIWYG editor
- ▶ Editing existing content and adding an image
- ▶ Adjusting the tabbed settings for content nodes
- ▶ Configuring comments
- ▶ Publishing an RSS feed of basic pages

Introduction

This chapter is about becoming familiar with creating, editing, and publishing content with Drupal. We will explore how to create content and add it into the menu structure. We will then build on this and see how to set up a WYSIWYG editor to allow rich text editing, and then we shall see how to upload images into the editor.

We will then look at the configuration options that are available to all content items, including looking at how to set up commenting. We finish with a recipe that shows us how to publish an RSS feed of a particular type of content.

Creating a basic page and adding it to the main menu

In this recipe we are going to see how to create a new **Basic page** item, and add it to the main menu.

How to do it...

1. Select **Content** from the admin menu; this loads the content administration panel:

The screenshot shows the TYPO3 content administration panel. At the top, there is a header with a '+ Add content' button. Below this is a 'SHOW ONLY ITEMS WHERE' section with dropdown menus for 'status' (set to 'any') and 'type' (set to 'any'), and a 'Filter' button. Underneath is an 'UPDATE OPTIONS' section with a dropdown menu set to 'Publish selected content' and an 'Update' button. The main area displays a list of content items in a table format. The columns are: TITLE, TYPE, AUTHOR, STATUS, UPDATED, and OPERATIONS. The items listed are:

OPERATIONS	UPDATED	STATUS	AUTHOR	TYPE	TITLE
edit delete	12/23/2011 - 15:06	published	dylan	News article	Sample news article
edit delete	12/21/2011 - 13:40	published	dylan	News article	November article
edit delete	12/13/2011 - 21:48	published	dylan	News article	Sample news content 2
edit delete	12/11/2011 - 18:22	published	dylan	News article	News item 1
edit delete	12/05/2011 - 00:01	published	dylan	Forum topic	Test topic

2. Select **+Add content**.

3. You will now see the list of available content types we can create content for.
Select **Basic page**.

The screenshot shows the configuration interface for a 'Basic page'. At the top, there is a 'Title *' field containing 'About us first draft'. Below it is a 'Body (Edit summary)' field with a large text area containing placeholder text about a diam nunc. Underneath the body is a 'Text format' section set to 'Filtered HTML' with a link to 'More information about text formats'. A list of allowed HTML tags is provided: Web page addresses and e-mail addresses turn into links automatically, Allowed HTML tags: <a> <cite> <blockquote> <code> <dl> <dt> <dd>, and Lines and paragraphs break automatically. On the left side, there is a vertical tab menu with options: 'Menu settings' (selected), 'Revision information', 'URL path settings', 'Comment settings', 'Authoring information', and 'Publishing options'. In the 'Menu settings' tab, there is a checkbox for 'Provide a menu link' which is checked, and a 'Menu link title' field containing 'About us'. Below that is a 'Description' field with the placeholder 'Click here to read all about us!' and a note 'Shown when hovering over the menu link.' At the bottom of the tab is a 'Parent item' dropdown menu showing '<Main menu>'.

4. Enter a title for your page, this is a required field.
5. Enter some body text for your page.
6. In the vertical tab section, under **Menu settings**, select the **Provide a menu link** checkbox to display the menu link options.
7. In **Menu link title** enter the text you would like to appear in the menu item.
8. In the **Description** field enter a small piece of descriptive text to appear in the menu item's tooltip.
9. For the **Parent item** option, leave **<Main menu>** selected.
10. Leave the **Weight** set as **0**.
11. Leave the other tab options in their default state.

12. Click on **Save**.
13. The display will change to the new page you created and you will now see your new menu item associated to this page in the main navigation menu:

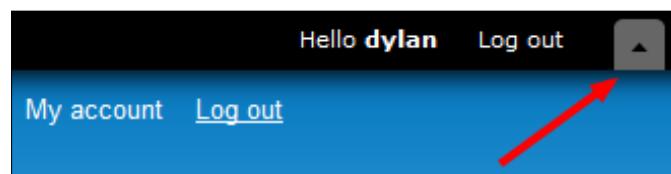


14. To change the order of the menu items, select **Structure** from the main menu.
15. Select **Menus**, followed by **list links** in the **Main menu** row.
16. Using the four-direction grab handle, drag the **Home** row above the **About us** row.
17. Click on **Save configuration**.
18. Close the overlay, and you will see that the main menu items are now in the correct order:



How it works

You can go directly to the **Add content** page by using the provided shortcut in the shortcut bar, which can be found by selecting the arrow on the top-right part of the dark toolbar, as in the following screenshot:



Shortcuts can be added to the shortcut bar from many different pages by selecting the **Add to default shortcuts link**, next to admin page title.

As you create more content types, the content type selection screen will grow. By default, Drupal ships with the very basic content types: **Basic page** and **Article**.

The **Basic page** type is intended for static content, or at least for content that changes infrequently. As with all content types in Drupal, they can be expanded to include other fields; for example, a downloadable file. This is discussed in detail in *Chapter 4, Custom Content Types*, where we will see how to create a content type for almost any use, such as a product, or a press release.

The body text field for **Basic page** doesn't let the user enter rich text using a WYSIWYG editor by default. This functionality needs to be installed.

The **Menu link title** is the text that will appear in the menu item; it will generally be the same as the title for the **Basic page**, but not necessarily. The **Description** text for the menu link appears as a tooltip when you hover the mouse over the menu item. This is optional, but it's useful for adding some supplementary information to a link.

In the **Parent item** drop-down you can select the menu item that will be the parent of your new **Basic page**. By default, there is only the **Main menu** and its associated items from which to choose. By leaving the option as **Main menu**, the page becomes a top-level menu item in the main menu.

The **Weight** option allows you to set where in the menu you would like the item to appear. It's an arbitrary value which lets you quickly set an approximate position for a menu item; for example, if you wanted a menu item to appear towards the start, you could set a negative number. In this case, we have opened the **Menu Editor** screen and manually updated the order of the menu items in the main menu, but the same effect could have been achieved by entering a number; for example, entering 10 into the **Weight** field, making the **About us** menu item *heavier*, and thus sinking further down the menu.

See also

- ▶ *Installing and configuring a WYSIWYG editor*
- ▶ *Adjusting the tabbed settings for content nodes*

Installing and configuring a WYSIWYG editor

In case you didn't know, **WYSIWYG** stands for **What You See Is What You Get!** It is generally used to refer to a rich text editor which is used to simplify the editing of HTML, much like the interface you would expect to see in a Word Processor. The standard Drupal 7 installation does not provide any means to compose or edit rich text. Drupal provides a framework for managing different text formats, and with the use of the WYSIWYG module, it is possible to install a variety of WYSIWYG editors, as we will see in this recipe.

Getting ready

For this recipe you will need the following:

- ▶ To download and enable the most-recent recommended release of the WYSIWYG module: <http://drupal.org/project/wysiwyg>
- ▶ Access your Drupal site through FTP or file access to your Drupal site to install **CKEditor**

How to do it...

We will begin by downloading the CKEditor. We will then configure the site to use the CKEditor, through the WYSIWYG module:

1. Select **Configuration** from the admin menu.
2. Under **Content authoring**, select **Wysiwyg profiles**.
3. Open the **download** link for **CKEditor** (in a separate tab if possible).
4. Download the latest release of the editor; at the time of writing this is **CKEditor 3.6.2**.
5. Extract the folder to your desktop.
6. Using your FTP client, create a new folder called `libraries` in your Drupal installation, if it doesn't already exist:
`Drupal root/sites/all/libraries`.
7. Upload your extracted `ckeditor` folder to your new `libraries` folder so that it reads:
`Drupal root/sites/all/libraries/ckeditor`.
8. When the upload is complete, go back to the **Content authoring** page in your Drupal site.
9. Refresh the page. You will be presented with a list of text input formats.

10. For **Full HTML** select the **CKEditor** option from the drop-down menu and click on **Save**:

A Wysiwyg profile is associated with an input format. A Wysiwyg profile defines which client-side editor is loaded with a particular input format, what buttons or themes are enabled for the editor, how the editor is displayed, and a few other editor-specific functions.

INPUT FORMAT	EDITOR	OPERATIONS
Filtered HTML	No editor	
Full HTML	CKEditor 3.6.2.7275	Edit Delete
Plain text	No editor	

To assign a different editor to a text format, click "delete" to remove the existing first.

Save

► INSTALLATION INSTRUCTIONS

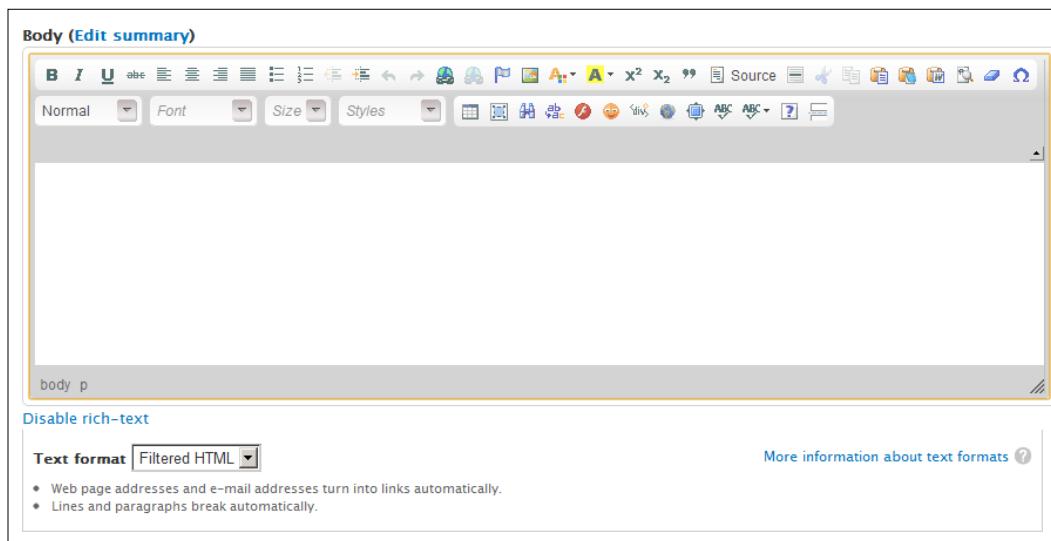
11. After saving, select **Edit** for **Full HTML**.
12. On the CKEditor configuration screen, open the **BUTTONS AND PLUGINS** field group.
13. You will see a list of all the available buttons for CKEditor. Check all of the items that you would like to include in your Full HTML editor.
14. Leave all other settings as default and click on **Save**.
15. Add a new piece of content and select **Full HTML** under **Text format**; you will see the editor load in the **Body** field:

Body (Edit summary)

Disable rich-text

Text format **Filtered HTML** More information about text formats ?

• Web page addresses and e-mail addresses turn into links automatically.
• Lines and paragraphs break automatically.



How it works...

You should now have a fully functioning WYSIWYG editor available. The editor can be used to edit all text areas and body fields.

When you are installing the module, you may have to enter your FTP details through the browser depending on how your Drupal installation is configured.

When uploading the `ckeditor` folder ensure that the folder inside the `libraries` folder reads `ckeditor`, and not `ckeditor_3.6.2/ckeditor`.

If you have an online file manager, such as cPanel, it is generally quicker to upload the CKEditor's ZIP file and extract it on the server.

There's more...

The basic WYSIWYG configuration will be satisfactory for most people, but if you want more functionality then there are a few extra things to consider.

Alternative WYSIWYG editors

You will notice that there are a few WYSIWYG editors to choose from such as TinyMCE, YUI Editor, and markItUp (useful for editing HTML source code), each with their own pros and cons; feel free to explore the features of the other editors.

Using the WYSIWYG for other text formats

We set CKEditor as the editor for **Full HTML** only, but it is entirely possible to set CKEditor as the editor for **Filtered HTML** or **Plain text**. This will depend on what options you would like to give to your users. For example, you could configure **Filtered HTML** to have a cut-down configuration of CKEditor with only cut and paste and strict XHTML compliance for less-trusted users, and for more capable users, you could configure the **Full HTML** with all of the CKEditor features.

Further settings for the WYSIWYG

We skipped over four of the configuration sections for the editor. Most of these features are self-explanatory. However, it may be useful to become familiar with the **Cleanup and Output** section.

Always ensure that **Verify HTML** is checked, as it can remove potentially malicious code that a user may knowingly or unknowingly paste into the editor.

Enable **Apply source formatting** if you would like the editor to layout your HTML in a readable fashion.

Enable **Force clean up on standard paste** to automatically remove some of the excessive markup and styles that are added to text when it is copied and pasted from applications such as Microsoft Word. Without this enabled, you run the risk of code that doesn't obey your theme's stylesheets, and a lot of extra markup. Alternatively, for more-trusted users, enable the **Paste from Word** function which allows the same functionality.

In the CSS section of the configuration page, you can select which CSS rules the editor will use to style its content in the **Editor CSS** option. For example, with **Use theme CSS** selected, the editor will attempt to display the content of the editor styled with the theme's rules.

In the **CSS options** field you can set additional styles that can be applied to editor text. This is useful if you need users to apply a custom class to an element, but you don't want them to apply it manually in the HTML view.

See also

- ▶ *Installing modules and themes* recipe in Chapter 1, *Installing and Configuring Drupal*
- ▶ <http://drupal.org/node/208456>: This link gives a comparison of WYSIWYG editors for Drupal

Editing existing content and adding an image

In this recipe, we will discover how to edit existing content, and then how to upload and insert an image into the editor. The WYSIWYG editor is already equipped with a button to add an image from a URL, but to upload an image directly to the server we will require a further two modules.

Getting ready

- ▶ You will need to have already installed the WYSIWYG editor as explained in the preceding recipe, *Installing and configuring a WYSIWYG editor*
- ▶ You will need to have the following modules downloaded, installed, and enabled:
 - http://drupal.org/project/imce_wysiwyg
 - <http://drupal.org/project/imce>



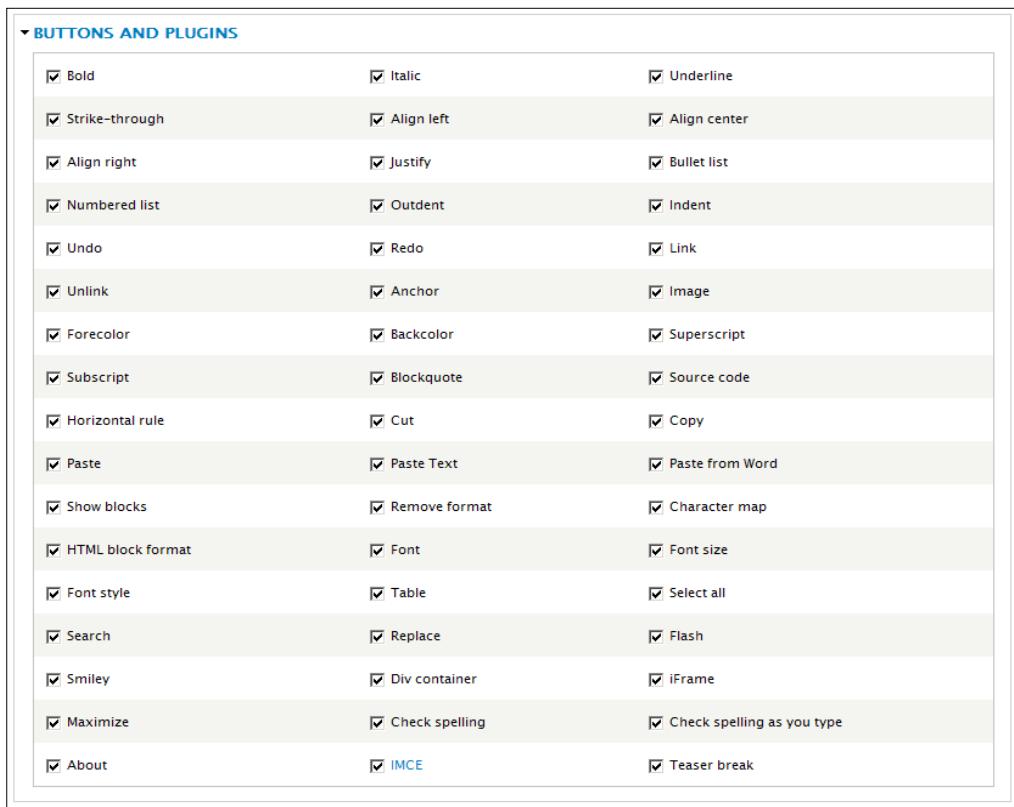
IMCE is a file manager that allows you to upload and manage images and other files into directories on the server. The IMCE WYSIWIG bridge module is responsible for providing IMCE support inside the WYSIWYG editor.

- ▶ Ensure you have at least one node of content in the content library of **Basic page** or **Article type**

How to do it...

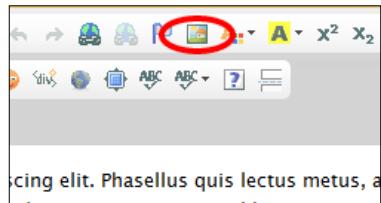
In the following steps we shall configure the **Full HTML** so that it can use the IMCE file uploading tool. We will then edit an item of content and add an image to it:

1. Select **Configuration** from the admin, then select **Wysiwyg profiles** under the **Content authoring** section.
2. Click on **edit** for the **Full HTML** text profile.
3. Open the **BUTTONS AND PLUGINS** field group, you will see that after installing the IMCE modules, there is an **IMCE** checkbox available:

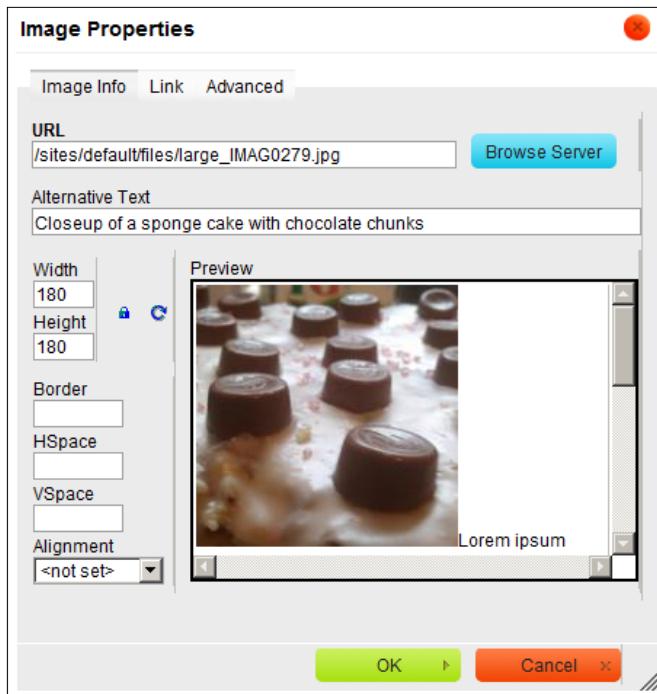


4. Select the **IMCE** checkbox and also ensure that the **Image** checkbox is checked; also enable any of the other features that you want.
5. Click on **Save**.
6. Select **Content** from the admin menu to open the content library.
7. Find the node of content you would like to edit and click on **Edit**.
8. Select the **Full text** format to load the WYSIWYG editor.

9. Select the image button in the editor:



10. In the **Image Properties** dialog box select **Browse Server** to open the file browser.
11. In the **File browser**, select **Upload** and then **Choose file** to open your OS's native file browser.
12. Find the image to use and click on **Open** which takes you back to the **File browser**.
13. In the **File browser** select **Upload** to transfer the file to the server.
14. You will now see your image in the **File browser**. Select the image and then select **Insert file** to use the image:



15. You will now see the **Image Properties** dialog with the new image loaded into it. Add some **Alternative Text** and click on **OK** to add the image to the editor.
16. Complete any editing you would like to do to the text and click on **Save** to finish.

How it works...

We begin by editing the WYSIWYG configuration options to enable the IMCE feature. The IMCE is what provides the ability of uploading images to the server.

After enabling the IMCE functionality, we edit an item of content and upload an image using the file browser.

There's more...

We have seen how to add images into content, but there are other ways to do this, and also other considerations such as the image's properties and size.

Image properties

When selecting an image to use in the editor, you can instead add an image from a URL. In the **Image Properties** dialog you can also set other options such as the image alignment, alt text, image link, CSS classes, and the image's CSS ID.

When uploading the image using the **File** browser, it is important to add **Alt text** (Alternative text). This is to convey the purpose of the image to visually impaired users who are accessing your site using a screen reader.

Image size

The image that you add to the editor may turn out to be too large. IMCE allows you to resample the image into a thumbnail. In the **File browser**, select the image to resize and then select **Thumbnails**. There are three resolutions to choose from by default (90x90, 120,120, 180x180), but you can add more through the IMCE configuration page.

You may also set the dimensions of the image in the **Image Properties** dialog box. Note that this does not resample the image, so any large images will still take the same time to load.

Alternative ways to add images to content

As we have just seen, one of the ways to add images to content is through the WYSIWYG editor. This may very well be enough for many use-cases, but what if you wanted to have more control over the image's resolution and position? It is probably not wise to trust an inexperienced content editor to add images of the right size and to ensure that the image has the correct class attached, and so on. This is where you would want to create a new content type with an image field where the output resolution and size can be automatically controlled, and the exact position in the content can be predetermined by the developer. All the content editor has to do is choose the image. See *Chapter 4, Custom Content Types* for more information.

Adjusting the tabbed settings for content nodes

In Drupal, content items are classed as nodes. Custom content nodes may have many different fields, but as they are all nodes they all share a selection of features, including the ability to have comment threads and content versioning. In this recipe, we will explore these shared features.

How to do it...

In this recipe, we will explore the various options that you may come across when editing a content node, such as providing a menu link, setting the node's URL path, and how to create a new revision:

1. Select **Content** from the admin menu and then select **+Add content**.
2. Select **Article** to create a new content node.
3. Add a **Title** for your content and the **Body** text.
4. Select the **Menu settings** tab, and check the **Provide a menu link** option.
5. In **Menu link title** enter the title as it will appear in the menu.
6. For **Parent item** select **<Main menu>**.
7. Select **URL path settings** and in the **URL alias** field enter the URL path you would like to use for the article, for example, `my-first-article`. Don't include any slashes.
8. Select **Comment settings** and ensure that **Open** is selected.
9. Select **Authoring information**. Update the **Authored on** field to January 1st 2012 by typing `2012-01-01`.
10. Select **Publishing options** and check **Published**, **Promoted to front page**, and **Sticky at top of lists**.
11. Click on **Save**.
12. You will now see your published article. Select the **Edit** tab to go back into the editing view.

About us

[View](#) [Edit](#) [Log](#)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis lectus metus, at posuere neque. Sed pharetra nibh eget orci convallis at posuere leo convallis. Sed blandit augue vitae augue scelerisque bibendum. Vivamus sit amet libero turpis, non venenatis urna. In blandit, odio convallis suscipit venenatis, ante ipsum cursus augue.

13. Update the body text by adding in a new sentence.
14. Select the **Revision information** tab, then check **Create new revision**.
15. Enter a piece of descriptive text into the revision log which summarizes the change you are making to the content, for example, "2nd draft of the article":

The screenshot shows the 'Revision information' tab settings for a content item. On the left, there is a sidebar with several tabs: 'Menu settings' (Not in menu), 'Revision information' (New revision), 'URL path settings' (No alias), 'Comment settings' (Open), 'Authoring information' (By dylan on 2011-10-31 22:44:49 +0000), and 'Publishing options' (Published, Promoted to front page). The 'Revision information' tab is active. On the right, there is a large text area labeled 'Revision log message' containing the text '2nd draft of the article'. Below this area is a note: 'Provide an explanation of the changes you are making. This will help other authors understand your motivations.' At the bottom of the form are three buttons: 'Save', 'Preview', and 'Delete'.

16. Click on **Save**; you will again see your published article, but now with some more body text.
17. You will also see a new **Revisions** tab next to the **Edit** tab; click on it.
18. You will now see the page of revisions of the content.
19. Select **revert** on the content with the earliest date and time.
20. Select the cross to close the revisions page; you will now see that the content has been reverted back to its state after the first save.

How it works...

Adding an URL alias for an item of content is said to be very important for SEO (Search Engine Optimization). It can also help to group areas of your site together; for example, if you have an "about" section, then the chances are that you would want all items belonging to this section to have a URL beginning with `about-us/`. If you do not enter anything for the URL alias then the content's path will be automatically generated and will take the form `node/x`, where `x` is a number.

Commenting can be turned on or off for each article. Commenting can be globally disabled for a content type if it's not ever going to be relevant.

The **Authored on** field stores a useful piece of information that can be used when constructing Views of data; for example, to create a list of news items which display in reverse chronological order. It's not the most user-friendly method of entering a date or time; just remember that the most significant number, the **year**, comes first, decreasing through to **second**, the least significant.

Menu settings About us	Authored by <input type="text" value="dylan"/> <input checked="" type="radio"/> Leave blank for <i>Anonymous</i> .
Revision information No revision	
URL path settings No alias	Authored on <input type="text" value="2012-01-01 00:00:00 +0000"/>
Comment settings Closed	Format: <i>2012-01-01 00:00:00 +0000</i> . The date format is YYYY-MM-DD and +0000 is the time zone offset from UTC. Leave blank to use the time of form submission.
Authoring information By dylan on 2012-01-01 00:00:00 +0000	
Publishing options Published	

The publishing options help to determine if and where a content node will be displayed. Selecting **Promote to front page** will cause the content to appear on the default home page, which displays content items in a blog style. If you are using a custom home page, then this option is not relevant. Selecting **Sticky at top of lists** is used for constructing Views where you want the content to always appear at the top of a list in order to get noticed.

Creating a revision is useful as it adds the ability to undo changes to content. An item of content can sometimes go through many changes before it is ready for publication. Adding a revision log message can help editors and publishers check what stage a piece of content is at, and also to assist in returning an item of content to a previous state. The log is very useful for conveying the intention of the author.

There's more...

You will now have grounding in the basic options that are available to all content nodes. This is very useful, but you will at some point require more advanced features for your content, there will also be situations where you will need more specific control over which of these features are permissible for a content type.

Pathauto

The Pathauto module is a very useful module for automatically generating URLs for content. For example, it can be configured so that all articles are given the URL `article/article-name`. When Pathauto is installed, it updates the URL path settings tab on a content edit page to automatically receive a default path according to the Pathauto rule for that content type, if one exists.

Granularity

It is possible to globally configure the availability of the revision and commenting options for all nodes created using a content type. For example, you may want to disable all commenting for all basic pages, or to enforce the system to create a new revision every time a node is saved. Furthermore, you can choose which user roles have the power to override the defaults. The configuration options for these settings are located in the content type edit page.

See also

- ▶ *Creating a more advanced content type recipe in Chapter 4, Custom Content Types*

Configuring comments

In Drupal, it is possible to permit commenting on every content node. Throughout this recipe we will see how to configure comments for the article content type. Following this we will create an article, add a comment and finally, moderate the comment.

Getting ready

Before we begin, ensure that the Comment module is enabled (the Comment module is part of the Drupal core module list).

How to do it

1. Select **Structure** followed by **Content types** from the admin menu.
2. Click on **edit** for the **Article** content type.
3. Select **Comment settings** from the tabbed options; ensure that the **Default comment setting for new content** is set to **Open**.
4. Leave the **Threading** checkbox checked.
5. Leave **Comments per page** set at **50**, and ensure that the two checkboxes **Allow comment title**, and **Show reply form on the same page as comments** are checked.
6. Leave **Preview comment** set to optional.
7. Click on **Save content type**.

8. Select **Content** from the admin menu, followed by **+Add content**.
9. Select **Article** and enter a title and body text, then click on **Save**.
10. Scroll to the bottom of the article; in the section **Add new comment** enter some text for the **Subject** and **Comment** fields.
11. Click on **Save** and then select **Content**.
12. Select the **COMMENTS** tab.
13. Select the checkbox of the comment to be deleted.
14. Choose **Delete the selected comments** from the **UPDATE OPTIONS** fieldset.
15. Click on **Update** to finalize the delete of the selected comment.

Deleted 1 comment.

UPDATE OPTIONS

Delete the selected comments

	SUBJECT	AUTHOR	POSTED IN	UPDATED	OPERATIONS
<input type="checkbox"/>	Nice article	dylan	My new article	11/01/2011 – 21:55	edit
<input type="checkbox"/>	Keep up the good work	dylan	My new article	11/01/2011 – 21:41	edit
<input checked="" type="checkbox"/>	Good article	dylan	My new article	11/01/2011 – 21:40	edit

How it works...

When setting the **COMMENTS** settings for the **Article** content type, the settings are being configured for all article nodes. The **Threading** feature will nest comments which are replies to other comments rather than replies to the article. Threading further indents comments that are replies to replies, and so on.

After saving a new comment, admins and editors with sufficient permissions will be able to delete a comment when viewing it in the article. However, when there are numerous comments to moderate it will be preferable to view comments for all content nodes, all in one place. When in the comments listing page we selected **Delete the selected comments**, but there is a further option to **Unpublish the selected comments**. You may also edit the content of the comment from this screen.

There's more...

Now that you are familiar with setting up commenting and how to moderate comments, there are a few more tasks that are relevant to comments.

Comment permissions

On the permissions page (**People | Permissions**) it is possible to set a bespoke set of comment permissions for each user role. Here you can see the various permissions available to the comment module:

PERMISSION	ANONYMOUS USER	AUTHENTICATED USER	ADMINISTRATOR
Block			
Administer blocks	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Comment			
Administer comments and comment settings	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
View comments	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Post comments	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Skip comment approval	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Edit own comments	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

We saw earlier how comments appear in published comments lists, but if a user does not have the **Skip comment approval** permission, then any comment they post will go into the **Unapproved comments** list.

Sending notifications when a comment is received

If you set up comment approver roles, or some similar user role, it would be a good idea to automatically alert them as soon as there is a new comment to moderate. This is easy to do using the **Triggers** module. The Triggers module allows you to assign actions on various comment, node, system, taxonomy, and user events. You may assign actions such as blocking a user, or sending an e-mail. You are also able to create your own custom actions.

Expanding the comment fields

Comments, like nodes, can have custom fields added to them. This is managed through the content type editing screen. You may, for instance, want to provide the user with a field to upload a document along with their comment, or maybe just a field to add their location; the possibilities are endless.

See also

- ▶ <http://drupal.org/documentation/modules/comment>

Publishing an RSS feed of basic pages

Publishing RSS feeds is an important requirement of modern websites. Drupal publishes a site-wide RSS by default as part of the core that lists all of the content nodes configured to appear on the front page. In this recipe, we will see how to create a more specific RSS feed which outputs items from one content type.

Getting ready

For this recipe you will need:

- ▶ The **Views** module installed and enabled
- ▶ The **ctools** module installed and enabled
- ▶ At least two existing items of **Basic page** content



The Views module is an incredibly useful module which can be used to display lists of nodes, users, taxonomy terms, and other types of content. Views can be used to produce a wide variety of lists, from a list of images arranged in a gallery, to a product listing.

How to do it...

1. Select **Structure** from the admin menu.
2. Select **Views**, and then on the **Views** page, select **+Add new view**.
3. In the **View name** field enter **Basic page RSS feed**.
4. In the **Show** drop-down ensure that **Content** is selected.
5. In the **of type** field, select **Basic page**.
6. Leave the **Sorted by** field as **Newest first**.
7. Uncheck the **Create a page** checkbox.
8. Click on **Continue & edit**.
9. On the views edit page select **+Add** and then choose **Feed**.
10. Under **FEED SETTINGS** select the trailing slash in the **Path** setting.
11. Enter `basic-page-feed.xml` as the path to our new feed and click on **Apply**.
12. Click on **Save** on the **View Edit** page.

Creating and Publishing Content

13. Go to Drupal root/basic-page-feed.xml to view your new RSS feed:

The screenshot shows the 'Feed' configuration page for a view named 'Feed'. The page is divided into several sections:

- Feed details:** Includes a 'Display name' field set to 'Feed'.
- TITLE:** Title is set to 'None'.
- FORMAT:** Format is set to 'RSS Feed'.
- FIELDS:** A note states 'The selected style or row format does not utilize fields.'
- FILTER CRITERIA:** Content: Published (Yes) and Content: Type (= Basic page).
- SORT CRITERIA:** Content: Post date (desc).
- FEED SETTINGS:** Path is set to '/basic-page-feed.xml', Attach to is 'None', Access is 'Permission | View published content'.
- HEADER:** An 'add' button.
- FOOTER:** An 'add' button.
- PAGER:** Items to display: 'Display a specified number of items | 10 items'.
- Advanced:** Includes sections for 'CONTEXTUAL FILTERS', 'RELATIONSHIPS', 'NO RESULTS BEHAVIOR', 'EXPOSED FORM', and 'OTHER'.
- OTHER:** Machine Name: 'feed_1', Comment: 'No comment', Use aggregation: 'No', Query settings: 'Settings', Caching: 'None', CSS class: 'None', Theme: 'Information'.

How it works...

The basic RSS feed output that is generated for the whole site is too vague for some uses, so we are creating our own custom RSS feed using the **Views** module.

When creating a new view we choose to deselect the **Create a page** checkbox as we will not be using a standard HTML view of the page, although leaving it selected will not do any harm.

When we get to the **View Edit** page, we need to create a new display for our view which in this case is a **Feed display** type. This will output the view query results as RSS.

3

Working with Blocks

In this chapter we will cover:

- ▶ Adding a new block
- ▶ Creating a submenu block
- ▶ Creating a Superfish menu block
- ▶ Creating a block with Views
- ▶ Adding a new block region to a theme
- ▶ Creating a mega footer menu
- ▶ Conditional display of a block

Introduction

Blocks constitute one of the fundamental methods of display in Drupal. Throughout this chapter we will be exploring how to add simple content blocks. Following this we will use blocks to expand upon Drupal's menu system by creating a submenu block from the main menu items, and then we will see how we can enhance the main navigation so that it becomes a drop-down menu.

Blocks aren't limited to basic content and menus, and we shall introduce the power of views in creating a simple views block. We will also look at how we can expand on the available regions of a theme, enabling greater positioning flexibility. Following this, and armed with the knowledge of creating menu blocks, we will go about constructing a mega footer menu. Finally, we will focus on block visibility and see how we can limit the display of a block based on the current page.

Adding a new block

In this recipe, we will discover just how simple it is to create a new content block and how to position it in the left-hand side column.

Getting ready

In this recipe, we make use of the WYSIWYG editor configured according to the recipe *Installing and configuring a WYSIWYG*, in Chapter 2, *Creating and Publishing Content*. It is not essential to the task, but it will assist while entering the content of our new block.

This recipe also assumes that you have the Bartik theme enabled and set to default. If you are not using Bartik, simply select a different region for the block.

How to do it...

1. Select **Structure** from the admin menu, then select **Blocks**.
2. Select **+Add block**.
3. In the **Block description** field enter an administrative description of the block:

The screenshot shows the 'Add new Block' form. It includes fields for 'Block description' (containing 'What's new? block'), 'Block title' (containing 'What's new?'), and 'Block body'. The 'Block body' field contains a WYSIWYG editor toolbar and a rich text area with the text 'Today we are pleased to announce that we are now using Drupal blocks on our site.' Below the editor is a 'body p' class selector. At the bottom, there is a 'Text format' dropdown set to 'Full HTML', a note about automatic links and line breaks, and a link to 'More information about text formats'.

Block description *
What's new? block
A brief description of your block. Used on the [Blocks administration page](#).

Block title
What's new?
The title of the block as shown to the user.

Block body *

Today we are pleased to announce that we are now using Drupal blocks on our site.

body p

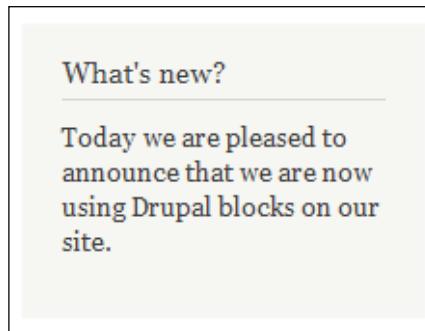
Disable rich-text

Text format Full HTML

• Web page addresses and e-mail addresses turn into links automatically.
• Lines and paragraphs break automatically.

More information about text formats

4. In the **Block title** field enter the title as it will appear above the block.
5. In the **Text format** drop-down, select **Full HTML**.
6. In **Block body** enter your content for the new block.
7. In **REGION SETTINGS** under the **Bartik** drop-down, select **Sidebar first** (or other similar region if not using the Bartik theme).
8. Leave the other options as they are and select **Save block**.
9. Go to your homepage and you will now see your new content block displayed in the left-hand side column:



How it works...

When creating the new block, we add an administrative description of the block that will be used to identify the block in the block listing page.

When entering the block's content we opt to use the **Full HTML**. This is a matter of user choice; you can just as easily enter the content without a WYSIWYG editor. As we are using HTML, it is possible to add virtually any type of content to the block including images and video.

When selecting the region where the block is to be added to, we only set a region for the current theme. You may have other themes installed, and it may be prudent to assign a region to the block for each theme, because otherwise your block will not display should you change your default theme.

After saving your new block you will arrive back at the blocks listing page. You can use this page to reorder the blocks in a particular position if there is more than one, or you can drag your block to a different position.

There's more...

By now you will be suitably adept at creating basic content blocks, but you're probably wondering how to add other types of blocks.

More block types

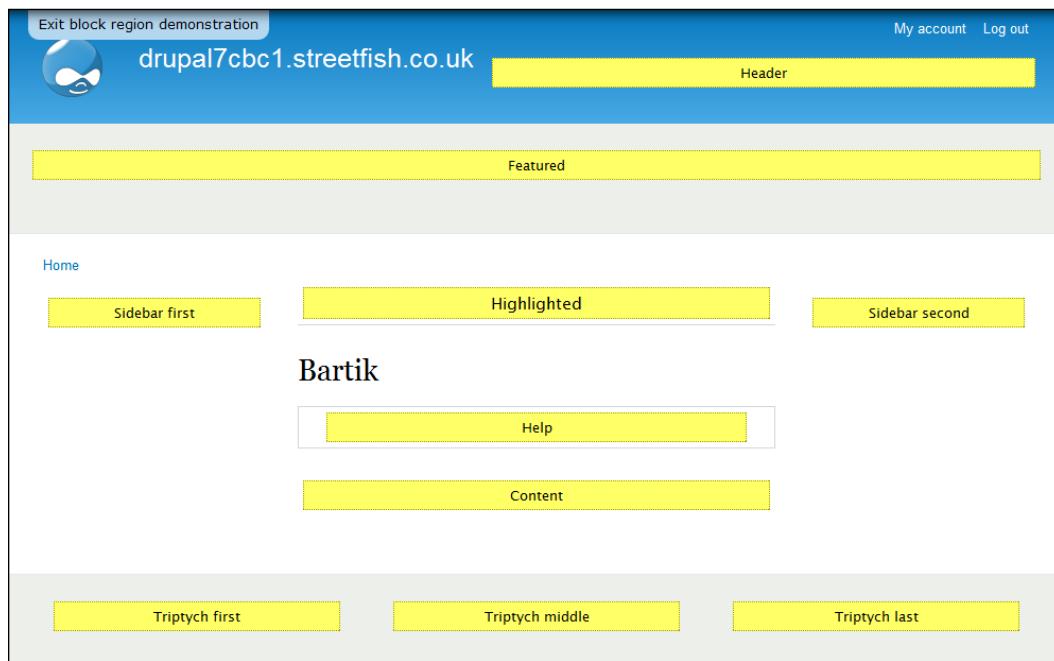
By default, you can only create static content blocks, but as we shall see later, it is possible to expand the repertoire of blocks that can be added using the **Menu** block module, which will allow you to add complex blocks created out of your main menu hierarchy. Another important block-creating tool is the **Views** module. Using this, you can create an unlimited range of block outputs based on your content types, from calendars to the latest forum posts.

Image blocks

The process for creating image blocks is almost identical to the way we have added text to a block in this recipe. The only difference is that instead of inputting text into WYSIWYG, you select the image tool and insert and upload an image instead.

Viewing the regions of a theme

To see a demonstration of the regions in a theme, go to the **Blocks** listing page, and select the option **Demonstrate block regions**.



See also

In this chapter:

- ▶ *Creating a submenu block*

In Chapter 5, *Using Views to Create Custom Lists, Grids, and Tables*:

- ▶ *Creating a latest news block*

Creating a submenu block

In this recipe, we will first populate our main menu with sample menu items by creating some basic page stubs. Following this, we will learn how to create a submenu block that will display a submenu generated from the main menu items.

Getting ready

For this recipe, you will need to install and activate the **Drupal Menu** block module:

http://drupal.org/project/menu_block.

This recipe also assumes that you have the Bartik theme installed and set to default. If you are not using Bartik, simply select a different region for your block in step 19.

How to do it...

1. Select **Add content** from the admin shortcut menu.
2. Select **Basic page**, then enter **About us** in the **Title** field.
3. In the **Menu settings** tab check **Provide a menu link**, and confirm that **About us** has been entered in the **Menu link title** field.
4. Ensure that **<Main menu>** is selected under **Parent item** and click on **Save**.
5. Select **Add content** and then select **Basic page**.
6. Enter **Contact us** in the **Title** field.
7. Select **Provide a menu link** and confirm that **Contact us** has been entered in the **Menu link title** field.
8. For **Parent item**, this time select **-About us**.
9. Click on **Save**.
10. Select **Add content** and then select **Basic page**.
11. Enter **Our history** in the **Title** field.

12. Select **Provide a menu link** and confirm that **Our history** has been entered in the **Menu link title** field.
13. For **Parent item** select **-About us**.
14. Select **Structure** from the admin menu, then select **Blocks**.
15. Select **+Add menu block**.
16. Enter **<none>** in the **Block title** field.
17. Enter **Submenu** in the **Administrative title** field.
18. Select **Main menu** in the **Menu** field.
19. For **Starting level** select **2nd level (secondary)**.
20. Under **Region settings** specify **Sidebar first** (or some other suitable region if not using the Bartik theme):

Basic options **Advanced options**

Block title

Override the default title for the block. Use `<none>` to display no title, or leave blank to use the default block title.

Administrative title

This title will be used administratively to identify this block. If blank, the regular title will be used.

Menu

The preferred menus used by `<the menu selected by the page>` can be customized on the [Menu block settings page](#).

Starting level

Blocks that start with the 1st level will always be visible. Blocks that start with the 2nd level or deeper will only be visible when the trail to the active menu item is in the block's tree.

Maximum depth

From the starting level, specify the maximum depth of the menu tree.

REGION SETTINGS
Specify in which themes and regions this block is displayed.

Bartik (default theme)

21. Click on **Save block** to finish.



How it works...

We begin this recipe by populating the main menu with a selection of basic pages, which we can use to test out our new menu block. When we select the **Parent item** for the node we are deciding on where in the menu that the node will be inserted. If you select a top-level menu item as the parent, then the node will be inserted as a child of the top-level item, and will be visible in the submenu block for that section of the site.

We create a new **Menu** block, which is the block type that we enabled by installing the **Menu** block module. The **Menu** block module provides the functionality for displaying vertical navigation menu trees.

When entering the details for the new menu block we enter <none> because in this case there's no need for a title for a submenu. We do, however, enter an administrative title that will be the *behind the scenes* name for the block as it will be displayed on the block listing page.

We select **Main menu** for the **Menu** field, but we could have selected any of the other menus in the system. We are simply deciding which menu to draw the submenu items from. You might, for example, want to define a footer menu; you could then create a footer menu block to display footer menu items for that section, in which case you would select **Footer menu** for the **Menu** field.

For the starting level we opt to display items from the second level; this is so that we only display the sub-level items for the section:

- ▶ Contact us
- ▶ Our history

If we were to set the starting level as the first level, then we would also see the parent item:

- ▶ About us
 - Contact us
 - Our history

This may in fact be what you want, in which case, go back and select the first level, but normally you would only want to display the subitems, as the parent will always be displayed in the main navigation.

You may be wondering about what happens with other main menu items, and their submenu items. As you add further items to the main menu, the submenu block will automatically output all of the available menu-items that are relevant to the current page, provided that the block is set to display on that page, which we will see more about later in the chapter on block visibility.

For the block's region we select **Sidebar first**; this is a sensible position for submenu navigation, but this is a matter of usability.

There's more...

You will now be armed with enough knowledge to build almost all of the menus you will need on a website, but what if we wanted to display a menu whose submenu items were fixed, regardless of what the parent page was? Or if you wanted to limit the depth of the submenu items?

Fixed starting item

In a menu with a fixed starting item, the parent link and its children are static (that is they don't update as you navigate through the site). You can create a fixed menu block as we have done in the recipe, but create it with the **Advanced options** tab selected. This provides the option of setting a **Fixed parent item**. Then it's simply a matter of selecting which item you want to keep as the fixed parent. This type of static configuration can be useful for building mega menus, or for surfacing a number of menus on a landing page.

Maximum depth

By setting the **Maximum depth** field, you can limit the depth of items that are shown in the menu, relative to the **Starting level**. This can be useful if your menu structure becomes quite deep, as the menu will start to get very messy, very quickly. You may also find that you are using a theme that doesn't support a menu-item depth past three or four levels.

See also

In this chapter:

- ▶ *Conditional display of a block*

Creating a Superfish menu block

The Superfish module unleashes the otherwise hidden submenu items in your main menu and automatically produces backward compatible drop-down menus for your existing theme. In this recipe, we will first install the required Superfish library, and then go on to create and configure a new Superfish menu block. Finally, we will need to disable the existing main menu.

Getting ready

To complete this recipe, you will need to install and enable the following modules:

- ▶ <http://drupal.org/project/libraries>
- ▶ <http://drupal.org/project/superfish>

You will also need to ensure that you are running jQuery 1.3.x or higher (this is included with the Drupal 7 core).

Also make sure that you have some main-menu items and some submenu items in your menu so we can demonstrate that the Superfish drop-downs are working.

In order to upload the Superfish library you will need FTP access to your site and an FTP client.

How to do it...

1. Go to the Superfish module page:
<http://drupal.org/project/superfish>
2. Find the link to the Superfish library and download it to your computer.
3. Extract the library ZIP file to your desktop.
4. Upload the extracted Superfish folder to your site under the following location:
Drupal root/sites/all/libraries. You may need to create the libraries folder if it doesn't already exist.
5. Select **Structure** from the admin menu and then select **Blocks**.
6. Find the block entitled **Superfish 1** and click on **Configure**.
7. In the **Block title** field enter **<none>**.
8. Ensure that **Menu parent** is set to **<Main menu>**.

Working with Blocks

9. Set the **Menu depth** to **1**.
10. Set the **Menu type** to **Horizontal**.
11. In the **Region settings** select **Header** for the Bartik theme (or another suitable region if not using the Bartik theme).

Block title

Override the default title for the block. Use <none> to display no title, or leave blank to use the default block title.

Menu Name

SUPERFISH SETTINGS

Menu Parent

The menu you want to be displayed using Superfish.

Menu Depth

The number of child levels starting with the parent selected above. **-1** means all of them, **0** means none of them.

Menu Type

Style

Animation speed

The speed of the animation either in **milliseconds** or pre-defined values (**slow**, **normal**, **fast**). (Default: fast)

12. Click on **Save block**.
13. Select **Structure** from the admin menu, then select **Menus**.
14. Select the **SETTINGS** tab.
15. For **Source for the main links** select **No Main links** from the drop-down.

16. Click on **Save configuration** to finish.



How it works...

When Superfish is installed and enabled, four blank Superfish blocks are created. We only need one of these to create our menu; you can reduce the number of Superfish that are available in the Superfish configuration page.

When configuring the block, we set the **Block title** to <none> because we don't normally want to display a title for a horizontal menu.

We set the **Menu depth** to **1** to avoid having too many confusing sub-level drop-down, which some users may find confusing and difficult to navigate.

We set the **Menu type** to **Horizontal**, but if we were making a sidebar menu we would set this to **Vertical**.

There are quite a few other options for configuring the menu, but we can leave the remainder of them in their default state.

In this example, we set the module position as **Header**; this may not be desirable for your production site, but serves as a good example of how we can position the new menu.

After activating the new Superfish block, we need to disable the existing main navigation block as it is no longer needed. We do this by setting the **Source for the main links** to **no main links**.

There's more...

We've seen how to add a new Superfish menu block to our site, but you will probably now want to replace the existing main navigation with the new Superfish block.

Replacing the main navigation with the Superfish block

It would be preferable to place the Superfish block in the same position as the existing main menu, but the Bartik theme does not provide a block region to do this. To overcome this problem you can create a subtheme of Bartik and add a new region such as **Navigation**.

It is necessary to create a subtheme of Bartik, rather than to directly modify the original theme itself because Bartik belongs to the core Drupal distribution. The core Drupal distribution should not be modified because it's likely to change when updating a Drupal installation.

There is a recipe later in this chapter that explains how to do this. Alternatively, you can avoid this problem by using a theme which already has a navigation region defined, such as Zen.

See also

In this chapter:

- ▶ *Adding a new block region to a theme*

Creating a block with Views

In this recipe, we will be creating a very simple block to display the five most recent items of content. To achieve this we will be using the **Views** module.

Getting ready

For this recipe you will need to install and enable the most recent recommended releases of the following modules:

- ▶ <http://drupal.org/project/views>
- ▶ <http://drupal.org/project/ctools>

You will also need at least five existing items of content (of any content type) for demonstrating the new block.

How to do it...

1. Select **Structure** from the admin menu and then select **Views**.
2. Select **+Add new view**.

The screenshot shows the 'Add new view' form. At the top, there is a field for 'View name *' containing 'Recent content listings'. Below it is a 'Machine name' field with the value 'recent_content_listings' and a link to '[Edit]'. There is also a checkbox for 'Description' which is unchecked.

The main configuration area has several sections:

- Show:** Set to 'Content' (selected), 'of type' 'All', and 'tagged with' (empty). 'sorted by' is set to 'Newest first'.
- Create a page:** Unchecked checkbox.
- Create a block:** Checked checkbox.
 - Block title:** 'Recent content listings'.
 - Display format:** 'HTML list' selected for 'of' and 'titles (linked)' selected for 'titles'.
 - Items per page:** '5'.

At the bottom are three buttons: 'Save & exit', 'Continue & edit' (highlighted in blue), and 'Cancel'.

3. In **View name** enter **Recent content listings**.
4. Ensure that the firstfieldset reads as follows: **Show Content of type All sorted by Newest first**.
5. Uncheck **Create a page**.
6. Check **Create a block**.
7. In the **Display format** field, select **HTML list** and ensure that **titles (linked)** is selected for the **of** field.
8. Leave the **Items per page** set as the default value of **5**.
9. Click on **Continue & edit**.
10. Select the link entitled **Full**, in the **Pager** section.
11. Select the option, **Display a specified number of items**.
12. Select **Apply (All displays)**.

13. In the following dialog box ensure that **Items to display** is set to **5** and click on **Apply (all displays)**.
14. You will now arrive back at the view's edit screen; click on **Save** to apply the pager configuration change.
15. Select **Structure** from the admin menu, and then select **Blocks**.
16. Scroll down to find the new block **View: Recent content listings**.
17. Select **Sidebar first** from the **REGION** drop-down (or any suitable region if not using the Bartik theme).
18. Scroll down and click on **Save blocks**.
19. Go to your homepage and you will now see your new block:



How it works...

In this recipe, we use the Views module to create a new block. The Views module is a powerful Drupal module which allows you to create dynamic lists of content based on various filter and sorting criteria.

The **Add new view screen** provides you with some shortcut options that assist in setting up the basis of your new view. For some very simple views you may not need to apply any more configurations. In this case, if we were to **Save and exit** with the options configured as they are, then we would have a block that contains pagination. This is not desirable for the very simple block we are creating, so we go to the view's edit page and remove the pagination.

At first glance, the view's edit page is rather overwhelming, but do not be put off. You will soon become familiar with all of the different options on the screen, and you'll be able to create a vast selection of blocks for your site. We will delve deeper into this subject later on in the book.

When a new view is created that contains a **Block** display type, a new block is added into the main blocks listing. This new block can then be positioned like any other block; it can also be imported into a custom panel page.

See also

Chapter 5, Using Views to Create Custom Lists, Grids, and Tables

Adding a new block region to a theme

A Drupal theme defines the structure of the HTML that is output to the screen. It also defines several regions in the HTML structure where blocks can be output, such as Header, Footer, Sidebar first, and so on. This recipe describes how to add your own custom regions to a theme, for situations where the predefined ones do not provide enough flexibility to display your content.

Getting ready

For this recipe, you will need:

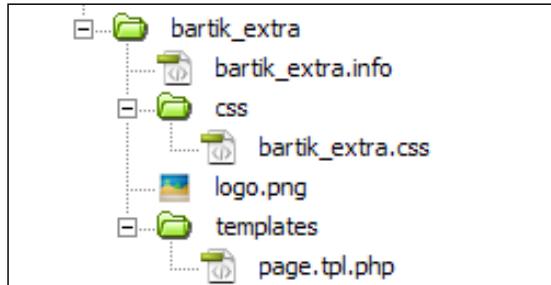
- ▶ FTP access to your site and a FTP client such as FileZilla
- ▶ A suitable code editor such as Notepad ++

How to do it...

In this recipe, we will be creating a subtheme based on the Bartik theme, to which we will add the new region. We will be working with a subtheme to avoid updating the Bartik theme directly, as it's not advisable to update any files from the core Drupal files, to prevent problems with overwriting when Drupal is updated.

1. Working locally, create a folder called `bartik_extra`.
2. In your `bartik_extra` folder, create a new file called `bartik_extra.info`.
3. In your `bartik_extra` folder create a sub-folder called `css`, and inside this, create a blank file called `bartik_extra.css`.
4. In your `bartik_extra` folder, create an empty folder called `templates`.
5. Log in to your FTP account for your site and navigate to `Drupal root/themes/bartik`.
6. Copy `logo.png` from `Drupal root/themes/bartik` to the root of your new `bartik_extra` folder.
7. Copy `page.tpl.php` from `Drupal root/themes/bartik/templates/` to your `bartik_extra/templates` folder.

8. Confirm that your new `bartik_extra` folder now has the following structure:



9. Open your new `bartik_extra.info` file, enter the following code, and save the file:

```
name = Bartik extra
base theme=bartik
description=A Bartik sub-theme which provides a navigation region
core = 7.x

stylesheets[all][] = css/bartik_extra.css

regions[header] = Header
regions[help] = Help
regions[page_top] = Page top
regions[page_bottom] = Page bottom
regions[highlighted] = Highlighted

regions[featured] = Featured
regions[content] = Content
regions[sidebar_first] = Sidebar first
regions[sidebar_second] = Sidebar second

regions[triptych_first] = Triptych first
regions[triptych_middle] = Triptych middle
regions[triptych_last] = Triptych last

regions[footer_firstcolumn] = Footer first column
regions[footer_secondcolumn] = Footer second column
regions[footer_thirdcolumn] = Footer third column
regions[footer_fourthcolumn] = Footer fourth column
regions[footer] = Footer

regions[navigation] = Navigation
```

10. Open your new CSS file `bartik_extra.css` and add the following style definition:
`.region-navigation{clear:both}`
11. Now open your copied version of the `page.tpl.php` in `bartik_extra/templates`.
12. Locate the line in the file where the header is outputted:
`<?php print render($page['header']); ?>`
13. Directly after this line, add the following line:
`<?php print render($page['navigation']); ?>`
14. Save the `page.tpl.php` file and upload your `bartik_extra` folder to `Drupal root/sites/all/themes/`.
15. Select **Configuration** from the admin menu and then select **Performance**.
16. Click on **Clear all caches**.
17. Select **Appearance** from the admin menu.
18. Locate the **Bartik extra theme**, and select **Enable and set as default**.
19. Your subtheme with the new navigation region is now ready to use.

How it works...

We create a subtheme of the Bartik theme, rather than just modifying the existing Bartik theme.



It is good practice not to update any of the core files and themes in a Drupal installation as it minimizes the risk of future updates to the core overwriting any changes you may have made. Remember that any files outside of the `files` directory are liable to be overwritten whenever the Drupal core is updated.

A Drupal subtheme will inherit all of the resources, CSS, and templates of the parent theme, but it won't inherit the `logo.png`, so we copy the parent's `logo.png` into our subtheme. A theme is not required to have a `logo.png` file, but we do this for aesthetic purposes.

When creating our subtheme, we include the required `.info` file, and any files we wish to override; in other words, files where we want to specify different functionality to the parent theme.

Working with Blocks

All Drupal themes must contain a `.info` file of the same name as the theme. In our new `info` file, `bartik_extra.info`, we first specify some of the metadata for the theme:

```
name = Bartik extra
base theme=bartik
description=A Bartik sub-theme which provides a navigation region
core = 7.x
```

For `name` we specify a unique human readable name for the theme, then for the `base theme`, we enter the theme name of the parent from which we want to inherit.

The `description` is an optional attribute, but is recommended. The `core` is a required attribute, and it indicates which versions of Drupal the theme is compatible with, in this case, just Drupal 7.x.

For the regions entries in the `info` file, all but one region is duplicated from the `bartik.info` file. We then add our new region below the parent's regions:

```
regions[navigation] = Navigation
```

We create the CSS file `bartik_extra.css` so that we can apply some styling that is specific to our subtheme. In this case we only add the following definition:

```
.region-navigation{clear:both}
```

This ensures that when Drupal outputs our new navigation region, it will output it beneath any header region items.

The `page.tpl.php` file is responsible for outputting the main layout markup for all pages. We want the subtheme to have exactly the same markup as the parent's, except with the addition of our new region. To do this we copy the original file and insert the following code, which tells Drupal where in the page's structure we want to output any navigation blocks:

```
<?php print render($page['navigation']); ?>
```

At the end of the recipe we refresh the site's cache so that we can be sure that Drupal is reading the latest version, and not a cached version of the new subtheme. Also be aware that you will have to refresh the cache after making any changes to the `.info` file. Finally, we enable the new subtheme.

There's more...

So far you will have created a new subtheme with a navigation region. That's not much use on its own, so it's likely you will want to experiment with placing some blocks in it.

Adding a Superfish menu block

Earlier on in this chapter we saw how to create a Superfish menu block, but replacing the main menu with the new Superfish block was not entirely possible with the Bartik theme as it didn't contain a navigation region. Now we have added the navigation region and can correctly place the Superfish menu block. In the blocks listing page, set the Superfish menu block's region to **Navigation**. After saving the configuration, the Superfish menu will now be in a more suitable location.

See also

In this chapter:

- ▶ *Creating a Superfish menu block*

In Chapter 9, *Creating Regular, Mobile, and Tablet Themes*:

- ▶ *Creating a new theme using Zen*

Creating a mega-footer menu

A mega-footer is a section of a website located above the main footer links which usually displays a menu listing for each subsection of the site. In this recipe, we will be creating a mega-footer consisting of four new menu blocks, each set to a different fixed parent item. We shall then add one menu block to each of the four footer columns in the Bartik theme.

Getting ready

For this recipe you will need:

- ▶ The Bartik theme, or a theme with at least four footer columns
- ▶ To install and enable the Menu block module:
http://drupal.org/project/menu_block
- ▶ To have at least four main menu items, each with at least four subitems

Working with Blocks

This recipe will be based upon the following menu structure, but feel free to use an alternative menu structure if you would prefer:

- ▶ Home
- ▶ About us
 - History
 - Our organization
 - Our people
- ▶ Contact us
 - E-mail us
 - Our locations
- ▶ Our portfolio
 - Portfolio A-Z
 - Portfolio archive
- ▶ What we offer
 - Business solutions
 - Independent advice
 - Solution consultation
 - Market research

How to do it...

1. Select **Structure** from the admin menu, then select **Blocks**.
2. Select **+Add new menu block**.
3. Select the **Advanced options** tab.
4. In the **Block title** field enter **About us**.
5. Set the **Maximum depth** to **1**.
6. For the **Fixed parent item** select **About us**.
7. In the **Region settings** select **Footer first column** for the **Bartik** theme.

8. Click on **Save block**.

The screenshot shows the 'Basic options' tab of a Drupal block configuration form. The tabs at the top are 'Basic options' (selected) and 'Advanced options'. The form fields include:

- Block title:** A text input field containing 'About us'. A note below it says: "Override the default title for the block. Use <none> to display no title, or leave blank to use the default block title."
- Administrative title:** An empty text input field. A note below it says: "This title will be used administratively to identify this block. If blank, the regular title will be used."
- Menu:** A dropdown menu set to 'Main menu'. A note below it says: "The preferred menus used by <the menu selected by the page> can be customized on the [Menu block settings page](#).
Starting level: A dropdown menu set to '1st level (primary)'. A note below it says: "Blocks that start with the 1st level will always be visible. Blocks that start with the 2nd level or deeper will only be visible when the trail to the active menu item is in the block's tree."
 Make the starting level follow the active menu item. A note below it says: "If the active menu item is deeper than the level specified above, the starting level will follow the active menu item. Otherwise, the starting level of the tree will remain fixed."
- Maximum depth:** A dropdown menu set to '1'. A note below it says: "From the starting level, specify the maximum depth of the menu tree."
 Expand all children of this tree.
 Sort menu tree by the active menu item's trail. A note below it says: "Sort each item in the active trail to the top of its level. When used on a deep or wide menu tree, the active menu item's children will be easier to see when the page is reloaded."
- Fixed parent item:** A dropdown menu set to '-- About us'.

9. Select **Structure** from the admin menu, then select **Blocks**.

10. Select **+Add new menu block**.

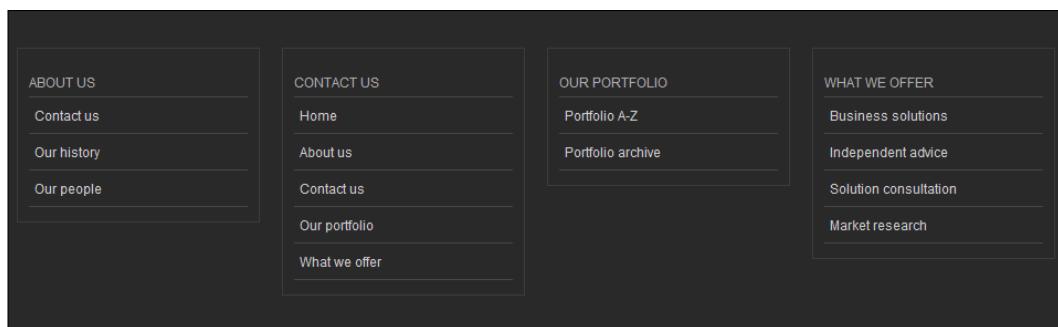
11. Select the **Advanced options tab**.

12. In the **Block title** field enter **Contact us**.

13. Set the **Maximum depth** to **1**.

14. For the **Fixed parent item** select **Contact us**.

15. In the **Region settings**, select **Footer second column** for the **Bartik** theme.
16. Click on **Save block**.
17. Select **Structure** from the admin menu, then select **Blocks**.
18. Select **+Add new menu block**.
19. Select the **Advanced options** tab.
20. In the **Block title** field enter **Our portfolio**.
21. Set the **Maximum depth** to **1**.
22. For the **Fixed parent item** select **Our portfolio**.
23. In the **Region settings** select **Footer third column** for the **Bartik** theme.
24. Click on **Save block**.
25. Select **Structure** from the admin menu, then select **Blocks**.
26. Select **+Add new menu block**.
27. Select the **Advanced options** tab.
28. In the **Block title** field enter **What we offer**.
29. Set the **Maximum depth** to **1**.
30. For the **Fixed parent item** select **What we offer**
31. In the **Region settings** select **Footer fourth column** for the **Bartik** theme.
32. Click on **Save block**.



How it works

When creating the new menu blocks we set the **Maximum depth** to **1**; this is because we only want to show one level of depth beneath the parent item. Our menu would become very cluttered if we were to show menu depths of two or more.

We set the **Fixed parent** value to the top-level item for the submenu. This will usually be one of your second level menu items, but there's no reason why you couldn't make a mega-footer which contains third level menu blocks.

The **Region settings** is where we select the region in the theme that we want to add the new block to. In this example, we are adding one block in each of the four footer regions of the Bartik theme:

- ▶ Footer first column
- ▶ Footer second column
- ▶ Footer third column
- ▶ Footer fourth column

There's more

This example only caters for a site with four top-level menu items. In the real world, you are likely to have a few more. Here are a few ways you can increase the number of blocks in your mega-footer.

Adding more blocks to one region

A quick way to add more menu blocks to your mega-footer would be by simply adding new blocks to one of the existing regions. This is quite a common approach, but depending on the size of each block, you may need to shuffle the blocks around to find a configuration that looks aesthetically balanced.

Adding more regions

When four columns are not enough, you might want to expand your theme to include five or maybe six footer column regions.

Adding other items

You don't have to stop at just adding menu blocks to your mega-footer. Many sites use this area for displaying content such as social media links and RSS subscription links.

See also

In this chapter:

- ▶ *Adding a new block region to a theme*

Conditional display of a block

So far in this chapter we have dealt with creating and positioning blocks. By default, blocks are set to display on every page, but in this recipe we will see how to configure the pre-existing **Who's new** block so that it displays only on the homepage, and only for logged in users.

How to do it...

1. Select **Structure** from the admin menu, then select **Blocks**.
 2. Find the **Who's new** block and select **configure**.
 3. Under the **Pages** tab, select **Only the listed pages**.
 4. In the text field enter **<front>**.
 5. Select the **Roles** tab.
 6. Check **authenticated user**, and **administrator**.
 7. Select **Save block**.

Visibility settings

Pages Restricted to certain pages	Show block on specific pages
Content types Not restricted	<input type="radio"/> All pages except those listed <input checked="" type="radio"/> Only the listed pages
Roles Not restricted	
Users Not customizable	

<front>

Specify pages by using their paths. Enter one path per line. The '*' character is a wildcard. Example paths are *blog* for the blog page and *blog/** for every personal blog. <front> is the front page.

Save block

How it works...

The block has now been configured so that it will only be visible when the user is viewing the homepage. We entered `<front>`, which is a shortcut for selecting the homepage as a criterion for displaying the block.

In the **Roles** tab we selected **authenticated user** and **administrator**. This adds further criteria that the page will have to meet in order to display the block. The page must now be the front page and the current user must be either an authenticated user, an administrator, or both.

There's more...

We now know how to display the block to particular user groups and how to control if a block appears only on the home page, or on all pages. But there are a few more methods of configuring block visibility that you should know about.

Visibility by content type

In the block's configuration page there is a tab called **Content** types. This lists the different content types available to the system. By selecting one or more of these content types you can configure the block to only display when a particular content type is being viewed.

Visibility by URL

In the recipe, we simply set the URL criteria so that the block only displays on the front page. But the URL field is very versatile and can be used to accomplish some customized behaviors. For example, if you wanted to display your block only for an entire section of your site, for example, /about-us/ then you could add the URL:

- ▶ /about-us/*

If you wanted to display your block for all of your /about-us/section and also on the home page, you could write the following:

- ▶ /about-us/*
- ▶ <front>

You can add as many URLs as necessary. The asterix is a wildcard and indicates that all of the sub-pages of /about-us/ are to trigger the display of the block.

Setting block visibility with other modules

The block visibility configuration is not limited to just the block's configuration page. You can use the Context module to configure sets of complex rules which can not only determine the visibility of blocks, but also what breadcrumb is displayed, and what menu item is set to be active.

Blocks can also be added to panels. Each item added to a panel has the option to add a visibility rule, which allows you to set criteria such as user role.

4

Custom Content Types

In this chapter we will cover:

- ▶ Creating a basic content type
- ▶ Configuring the output of the content type
- ▶ Applying an image format
- ▶ Installing more field types
- ▶ Creating a more advanced content type
- ▶ Building a custom content importer
- ▶ Building a forum

Introduction

This chapter is all about creating and configuring content types. In Drupal, all content is organized into nodes, and each node has a particular content type, such as Basic page, Article, or Blog entry, which ship with the Drupal core. What differentiates one content type from another, are their custom fields. For example, if you were making a job listings section of your site, you could create a content type, Job, which might have the additional fields, Salary, Hours of work, and Location.

There are many content types that you can create to improve the structure of your content; for example, News, Event, Venue, Product, or even Property if you were making a website for an estate agency.

Content types in Drupal, or any website, are useful not only because they allow the user to quickly add content to a site, but also prescribe a controlled structure to the output of that content, which is really good news for consistency.

We start off by creating a very simple content type. We then look at the configuration options for the output of a content type. Following this we see how we can create and apply image formats to further configure the output, and to assist in making content appear uniform. We then move on to installing new field types, which illustrates how we can expand on the variety of possible content types we can create. We then use the knowledge learnt from the start of the chapter to create a more complex content type that helps to solve a real world requirement of making a job listings site.

Towards the end of the chapter we see how it is possible to import content into a custom content type in a batch process, using the Feeds module. Finally, we end the chapter with a recipe for setting up and configuring the native Drupal Forum module.

Creating a basic content type

In this recipe, we will learn how to create a very simple content type called **Menu item**, which could be used in conjunction with Views to create a menu for a restaurant. To create the content type we will first set up the default fields; following this we will add a new field named **Price**.

Our **Menu** item content type will consist of the following fields:

- ▶ Title (Node module element)
- ▶ Body (Long text and summary)
- ▶ Price (Decimal)

How to do it...

In the following steps we will be creating a new content type called **Menu item**, then adding and configuring a decimal field called **price**:

1. Select **Structure** from the admin menu, then select **Content types**.
2. Select **+Add content type**.
3. In the **Name** field enter **Menu item**.
4. In the **Description** field enter **Use menu items for adding new dishes to your menu**.
5. Leave the other settings with their default values and select **Save and add fields**.
6. In the **Add new field** heading enter **Price** as the label.
7. In the **Field name** field enter **price**.

8. In the **Type of data to store** drop-down select **Decimal**:

+	Title	title	Node module element	
+	Body	body	Long text and summary	Text area with a summary edit delete
+	Add new field			
	Price	field_ price	Decimal	Text field
	Label	Field name (a-z, 0-9, _)	Type of data to store.	Form element to edit the data.
+	Add existing field			
		- Select an existing field -	- Select a widget -	Form element to edit the data.
	Label	Field to share		

9. Click on **Save**.

10. On the **Field settings** page leave the **Precision** as **10**.

11. Leave the **Scale** as **2** and leave the **Decimal marker** as **Decimal point**:

PRICE FIELD SETTINGS

These settings apply to the *Price* field everywhere it is used.

Number of values
1

Maximum number of values users can enter for this field.
'Unlimited' will provide an 'Add more' button so the users can add as many values as they like.

Precision
10

The total number of digits to store in the database, including those to the right of the decimal.

Scale
2

The number of digits to the right of the decimal.

Decimal marker
Decimal point

The character users will input to mark the decimal point in forms.

12. Click on **Save field** settings.
13. On the following **Settings** page, check the checkbox labeled **Required field**.
14. In the **Help text** field, enter **Enter the item's price, do not add a currency symbol as this is added automatically**.
15. In the **Prefix** field, enter the following symbol: **£**.
16. Ensure that the field **Number of values** is set to **1**.
17. Click on **Save settings**.
18. Your new content type is ready to use.

How it works...

We begin by entering the title of the content type and a description. The description is optional, but can be very useful for content authors.

On the following page we see the list of fields associated with our new **Menu item** type. We then add our new field name, which will be the human readable name for the field, and a unique machine name for the field, which will be used internally by the system and used in templates. We then select the data type for the field, in this case, decimal, as we will be using the field to display numbers with two decimal places.

After we add a new field, it needs to be configured. So, on the first field settings page we set the **Precision** as **10**, which indicates that we want our number to be limited to ten digits, including decimal places. We then set the **Scale** to **2**, which will enforce the rule that the number will have two decimal places.

On the following **Settings** page we have a number of different options; first, we check the checkbox to make the field a **Required field**. This ensures that the field is not left empty, as we know that every item on a menu will require a price.

We then enter **Help text**, which assists users entering data with some useful hints about the format of the data they will need to enter. We skip past the fields for **Minimum** and **Maximum**, which could be used to restrict the price to a defined range, but if you know that your price will have a particular range then these fields give you the opportunity to do that.

We insert the pound symbol (£) in the **Prefix** field so that the prices will be output with the currency symbol automatically displayed before the value. You can also enter a **Suffix** in all, making the decimal data type very flexible.

We leave the field **Number of values** set to **1**; this means that we only ever want the user to add one price for a menu item.

There's more...

In this example we have made a very simple content type, which doesn't really do a lot on its own, but serves to illustrate the ease at which a new content type can be created. However, with a few extra steps you can put the **Menu item** content type to work in a fully-fledged menu page.

Making a Menu view

To create a menu system for a restaurant, you already have the content type with which to add and edit menu items. The next step is to create a new view using the Views module. When setting up the view, set it to display content of the **Menu item** type.

See also

In Chapter 5, *Using Views to Create Custom Lists, Grids, and Tables*:

- ▶ *Creating a news listing view*

Configuring the output of a content type

A new content type, by default, will output all of its fields in the order in which they exist in the **Manage** fields tab. In this recipe, we will modify the order of the output fields in the **Article** content type and also change the way that the labels are displayed and finally see how we can apply some formatting options.

How to do it...

To begin, we will modify the order of the output fields, then change the **hide the tags** label, and finally we will update the **Teaser** view so it only displays a teaser of 400 characters:

1. Select **Structure** from the admin menu, then select **Content types**.
2. Select **Manage display** for the **Article** content type.
3. Drag the **Image** field below the **Body** field.

Custom Content Types

4. In the **Tags** row, in the **Label** column, set the drop-down value to <Hidden>:

FIELD	LABEL	FORMAT
Body	<Hidden>	Default
Image	<Hidden>	Image
Tags	<Hidden>	Link

5. Click on **Save**.
6. Select the **Teaser** button just below the tabs.
7. In the **Body** row, select the configuration cog.
8. In the **Trim length** field, update the default value to **400**.
9. Click on **Update** and then on **Save**.

How it works...

The **Manage** display page lists all of the fields associated with the content type. This page allows you to rearrange the order of display of all fields. In this example, we have moved the **Image** below the **Body**; this will cause the image to output below the body text, in the default view mode.

The **Manage** display page also provides the ability to configure the formatting of the label. You can choose to hide the label of a field completely, which is what we have done for the **Tags** field. It's also possible to output the label **Inline**, on the same line as the field content, or **Above**, which sets the field content to output on the line below the label.

The **Format** drop-down provides varying numbers of options depending on the field type. For example, the **Link** field has three formatting options, **Link**, **Plain text**, and **<Hidden>**. These options determine whether the field should be outputted as a link, or just the text with no link, or whether the field should be hidden completely.

Each content type has at least two view modes, **Default** and **Teaser**. The **Default** view mode options are used for configuring the display of the content type when it's being viewed in its node view context. The **Teaser** view mode is the context where the node is being viewed as part of a list view, such as a list of blog articles. This will usually be a shortened version of the full node, and in our example we have set the body text to only display its first 400 characters.

See also

In this chapter:

- ▶ *Applying an image format*

Applying an image format

Drupal 7 have a very useful image-formatting tool which automatically renders images at a predefined resolution. In other words, for source images uploaded in an image field, the image format can create image derivatives at the resolution specified in the format, leaving the source image intact. If the source image is then modified, the derivative images are reconstructed as needed.

For example, you might want to always display a user's profile picture at a particular resolution on their profile. Using an image format, you can guarantee that whatever image the user uploads, the profile picture will be at the resolution you want.

In this recipe, we will be creating a new image format for the **Article** content type and applying it so that all images displayed in the article's node view are rendered at 150x50 pixels.

How to do it...

We will first create a new **Image** format; following this we will be applying the new image format to the **Article** content type:

1. Select **Configuration** from the admin menu.
2. Select **Image styles** from the **Media** section.

3. Select **+Add style**.
4. Enter **article** in the **Style name** field.
5. On the **Edit style** page, select **Scale and crop** from the **EFFECT** drop-down.
6. Click on **Add**.
7. In the **Width** field enter **150**, in the **Height** field enter **50**.
8. Click on **Add effect**.
9. Select **Update style** from the **Edit style** page.
10. Select **Structure** from the admin menu.
11. Select **Content types**, then select **Manage display** for the **article** content type.
12. Select the gear for the **Image** field.
13. From the **Image style** drop-down select **Article**.
14. Leave the **Link image to** field set as **Nothing**.
15. Click on the **Update** button in the **Image** field.
16. Click on **Save**.

Show row weights		
FIELD	LABEL	FORMAT
⊕ Body	<Hidden>	Default
⊕ Image	<Hidden>	Format settings: Image Image style article Link image to Nothing

Update Cancel

How it works...

We start off by creating a new image format style. To do this we enter a name for the new format. The name must be all lowercase alphanumeric characters, without spaces. You can use hyphens or underscores for spacing.

After creating the new style, we apply the affects. There are six effects that can be added by default. You can also apply multiple effects. We only apply the **Scale and crop** effect. This will generate a derivative image at the selected size and crops any excess areas of the image.

After creating and configuring the new image style we then update the Article's **Image** field so that it uses the new **article** image style. Once the Article's **Display configuration** is saved, all images belonging to an article will be displayed scaled and cropped to a dimension of 150x50 pixels.

There's more...

We have seen how to apply a scale and crop an image, but that's not all that the **Image** formats can do.

Applying other effects

In addition to the scale and crop effect, Drupal 7 provides the ability to render images in black and white using the **Desaturate** effect. It is possible to apply rotations to your images using the **Rotate** effect. You can also apply a more advanced **Crop** effect where you can enter a crop dimension, and specify a starting point for the crop, for example, top-left or centre.

If the Drupal image effects aren't enough, then you can install third-party effects. For example, the **ImageMagick Raw Effect** module allows you to specify your own ImageMagick commands for the image format.

Manual cropping

In this recipe we have applied an automatic image crop effect. This is very useful, but you may require more control over which portion of the image is used when the image is cropped. Using a module such as the EPSA Crop module, you can create a new effect where the dimensions are constrained, but you can also decide, at the time of upload, which section of the image you would like to use in the crop.

See also

- ▶ The ImageMagick Raw Effect module: http://drupal.org/project/im_raw
- ▶ The EPSA Crop module: <http://drupal.org/project/epsacrop>

Installing more field types

Drupal provides many useful field types which can be used for building custom content types such as Image, Integer, Term reference, and File. However, it doesn't end there. In this recipe we will add a new file type, Link, to the site. Following this we will add a **Link** field to the **Basic page** content type.

Getting ready

For this recipe you need to install and enable the Link module:
<http://drupal.org/project/link>.

How to do it...

In this recipe we will first add a **Link** field to the **Basic page** content type. We will then create a new **Basic page** and add two sample links to it:

1. Select **Structure** from the admin menu, then select **Content types**.
2. Select **Manage fields** for **Basic page**.
3. In the **Add new field** section, enter **Related links** for the **Label** of the new field.
4. Enter **basic_page_links** for the **Field name**.
5. In the **FIELD** drop-down, select **Link**:

The screenshot shows the 'Add new field' configuration screen. It has a 'Label' input field containing 'Related links', a 'Field name' input field containing 'basic_page_link', and a 'Type of data to store' dropdown set to 'Link'. Below the form, a note reads 'Form element to edit the data.'

6. Click on **Save**.
7. On the **FIELD SETTINGS** page, select **Save field settings**.
8. On the **BASIC FIELD SETTINGS** page, ensure that **Required field** is not checked.
9. In the **Help text** field add the text **Enter a title for the link followed by a URL.**
If you are entering an external link check Open URL in a New Window.
10. Ensure that **Validate URL** is checked.
11. Do not check **Optional URL**.
12. For the **Link Title**, leave **Optional Title** selected.
13. Leave the **Max length of title** field, and **URL Display Cutoff** fields with their default values.

14. For the **Link Target**, select **Allow the user to choose**:

Link Title

Optional Title
 Required Title
 Static Title
 No Title

If the link title is optional or required, a field will be displayed to the end user. If the link title is static, the link will always use the same title. If [token module](#) is installed, the static title value may use any other node field as its value. Static and token-based titles may include most inline XHTML tags such as *strong*, *em*, *img*, *span*, etc.

Static title

This title will always be used if "Static Title" is selected above.

Max length of title field

Set a maximum length on the title field (applies only if Link Title is optional or required). The maximum limit is 255 characters.

URL Display Cutoff

If the user does not include a title for this link, the URL will be used as the title. When should the link title be trimmed and finished with an ellipsis (...)? Leave blank for no limit.

Link Target

Default (no target attribute)
 Open link in window root
 Open link in new window
 Allow the user to choose

15. In the **Number of values** field select **Unlimited**.
16. Click on **Save settings**.
17. Select **Add content** from the admin menu.
18. Select **Basic page**.
19. Add a **Title** for the page and some sample **Body** text.
20. In the **Related links** section enter **Google** in the **Title** field and enter <http://www.google.com> in the **URL** field.

21. Check **Open URL in a New Window**.
22. Select **Add another item**, then add another link of your choice.
23. Click on **Save** to finish.

Page with related links

[View](#) [Edit](#)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis lectus metus, at posuere neque. Sed pharetra nibh eget orci convallis at posuere leo convallis. Sed blandit augue vitae augue scelerisque bibendum. Vivamus sit amet libero turpis, non venenatis urna. In blandit, odio convallis suscipit venenatis, ante ipsum cursus augue.

Et mollis nunc diam eget sapien. Nulla facilisi. Etiam feugiat imperdiet rhoncus. Sed suscipit bibendum enim, sed volutpat tortor malesuada non. Morbi fringilla dui non purus porttitor mattis. Suspendisse quis vulputate risus. Phasellus erat velit, sagittis sed varius volutpat, placerat nec urna. Nam eu metus vitae dolor fringilla feugiat. Nulla.

Related links:

[Google](#)
[Wikipedia](#)
[Bing maps](#)

How it works...

When we name the new **Link** field, we need to enter a machine name for the field without spaces. The field name we use is **basic_page_links**. We could have entered just **links**, but this could cause ambiguity and conflict if you later try to create other content types with a **Link** field.

After opting to add the new **Link** field we arrive at the **Field settings** page; in this case, there are no further settings for the field, so we continue.

On the **Basic page settings** page, we don't choose to make the field a **Required field** because we want to make it optional to add links to the page.

We add some help text for the field to assist the user with filling out the field. This is a good chance to make the user aware of any editorial or style restrictions you want them to follow.

We check the **Validate URL** option so that we can be sure that the user has entered a valid URL; this helps to reduce the chance of the user entering broken links.

We don't check **Optional URL** as we want to ensure that the user *does* enter a URL when adding a new link; this prevents the user from forgetting to add the URL in.

For the **Link title**, we let the user decide whether they want to add a URL title, but in some circumstances it may be useful to enter a **Static title**. For example, if you were providing a field to link to an external map of a location, you could enter a **Static title** such as **View on a map**. This would help to provide consistency throughout your content.

It is generally desirable for external links to open in a new window to prevent users from navigating away from your site; it is also desirable for internal links to open in the same window. By setting the **Link target** to **User chooses**, we leave this option for the content inpuuter to decide. If we knew that this particular field would always be used for external links, then it would be better to select **Open link in new window**.

After saving the content type we then move on to creating an item of example content to test the new **Link** field. When adding the link fields, note it is possible to rearrange the order of the links by using the drag handle at the start of the row.

There's more...

In this recipe, we have added a new field type and used it in an existing content type. It is well worth exploring the range of field types that are available to add to your site, but there is one very useful field type that hasn't yet been mentioned.

Field collection

The Field collection module enables the creation of a new field type using existing field types. For example, if you were building a content type called Recipe, for a cookery website, you could create a new **Field collection** called Ingredients in which you could have a text field for the ingredient name, a text field for the quantity, and a drop-down for the units. The user would then be able to add any number of ingredients to the recipe while ensuring that the formatting is consistent throughout the site.

See also

- ▶ The Field collection module: http://drupal.org/project/field_collection

Creating a more advanced content type

In Drupal, unlike in many other CMSs, it is possible to create a vast range of content types for almost any requirement. In this example, we will be creating a new content type called Job. This content type can then be used to create a job posting system for your site.

Our Job content type will consist of the following fields:

- ▶ Title (Node module element)
- ▶ Body (Long text and summary)
- ▶ Salary (Text)
- ▶ Closing date (Date)
- ▶ Related documents (File)
- ▶ Position type (List)

Getting ready

For this recipe you will need to install the Date module:

<http://drupal.org/project/date>

You will then need to enable the following features of the Date module:

- ▶ Date
- ▶ Date API
- ▶ Date popup

How to do it...

Through the course of this recipe we will be creating a new content type with fields to add the job description, salary, closing date, related documents to download, and a field to select whether it's a permanent position:

1. Select **Structure** from the admin menu, then select **Content types**.
2. Select **+Add content type**.
3. Enter **Job** in the **Name** field.
4. In the **machine-readable name** field enter **job**.
5. In the **Description** field enter **Use the Job content type to create new job posts**.
6. Click on **Save and add fields**.
7. In the **Add new field** section enter **Salary** in the **Label** field.
8. In the **Field name** field enter **job_salary**.
9. In the **Field** drop-down select the **Text** field type.
10. Click on **Save**.

11. On the **Field settings** page, leave the **Maximum length** as **255** and select **Save field settings**.
12. On the **Job settings** page ensure that the **Number of values** is set as **1**.
13. Click on **Save settings**.
14. In the **Add new field** section enter **Closing date** in the **Label** field.
15. In the **Field name** field enter **job_closing_date**.
16. In the **Field** drop-down select the **Date** field type.
17. In the **Widget** drop-down select the **Pop-up calendar** widget.
18. On the **Field settings** page deselect the **Hour** and **Minute** fields for **Date attributes to collect**.
19. Ensure that **Collect an end date** is not checked.
20. Ensure that **Site's time zone** is selected for **Time zone handling**.
21. Click on **Save field settings**.
22. On the **Job settings** page, enter the following text in the **Help text** textarea: **Select the closing date for applications using the date popup**.
23. Click on **Save settings**.
24. In the **Add new field** section, enter **Related documents** in the **Label** field.
25. In the **Field name** field enter **job_related_documents**.
26. In the **Field** drop-down select the **File** field type.
27. Click on **Save**.
28. On the **Field settings** page, leave the **Enable Display** field and **Files displayed by default** checkboxes deselected.
29. Ensure that **Public files** is selected for the **Upload destination**.
30. Select **Save field settings**.
31. On the **Job settings** page, enter the following text in the **Help text** field: **Use this field to upload related documents such as application forms, job descriptions and candidate profiles**.
32. In the **Allowed file extensions** field remove **.txt** and add the following text: **PDF, DOC, DOCX**.

Custom Content Types

33. In the **File directory** field enter **job_related_documents**:

JOB SETTINGS

These settings apply only to the *Related documents* field when used in the *Job* type.

Label *
Related documents

Required field

Help text
Use this field to upload related documents such as application forms, job descriptions and candidate profiles.

Instructions to present to the user below this field on the editing form.
Allowed HTML tags: <a> <big> <code> <i> <ins> <pre> <q> <small> <sub> <sup> <tt> <p>

Allowed file extensions *
pdf, docx, doc

Separate extensions with a space or comma and do not include the leading dot.

File directory
job_related_documents

Optional subdirectory within the upload destination where files will be stored. Do not include preceding or trailing slashes.

34. In the **Number of values** field select **Unlimited**.
35. Click on **Save settings**.
36. In the **Add new field** section enter **Position type**.
37. In the **Field name** field enter **job_position_type**.
38. In the **FIELD** drop-down select the **List (text)** field type.
39. In the **WIDGET** drop-down select the **Check boxes/radio buttons** widget.
40. Click on **Save**.

41. On the **Field settings** page enter the following text: **permanent|Permanent temporary|temporary** in the **Allowed values list** field:

POSITION TYPE FIELD SETTINGS

These settings apply to the *Position type* field everywhere it is used. Because the field already has data, some settings can no longer be changed.

Number of values
 Maximum number of values users can enter for this field.

Allowed values list
 permanent|Permanent
 temporary|temporary

42. Click on **Save field settings**.

43. On the **Job settings** page ensure that **1** is selected for the **Number of values** field.

44. Click on **Save settings**.

LABEL	NAME	FIELD	WIDGET	OPERATIONS
⊕ Title	title	Node module element		
⊕ Body	body	Long text and summary	Text area with a summary	edit delete
⊕ Salary	field_job_salary	Text	Text field	edit delete
⊕ Closing date	field_closing_date	Date	Pop-up calendar	edit delete
⊕ Related documents	field_related_documents	File	File	edit delete
⊕ Position type	field_job_position_type	List (text)	Check boxes/radio buttons	edit delete
<hr/>				
Add new field <input type="text"/> Label				
field_ <input type="text"/> Field name (a-z, 0-9, _)		- Select a field type - <input type="button" value="▼"/> Type of data to store.	- Select a widget - <input type="button" value="▼"/> Form element to edit the data.	

45. Select **Manage display**.
46. Drag the **Body** field to the bottom of the list.
47. Drag the **Related documents** field to the position directly above the **Body** field.
48. For the **Salary**, **Closing date**, **Position type**, and **Related documents** fields select **Inline** from the **Label** drop-down.
49. Click on **Save**.

Data entry clerk

[View](#) [Edit](#)

Submitted by dylan on Fri, 12/02/2011 - 19:17

Salary: £20,000

Closing date: Tuesday, December 20, 2011 - 19:15

Position type: Permanent

Related documents: [JobDescription.docx](#)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis lectus metus, at posuere neque. Sed pharetra nibh eget orci convallis at posuere leo convallis. Sed blandit augue vitae augue scelerisque bibendum. Vivamus sit amet libero turpis, non venenatis urna. In blandit, odio convallis suscipit venenatis, ante ipsum cursus augue.

How it works...

In this example we add a variety of different field types. First we add the **Salary** field, which is added as a textfield type. For the job field we limit its length to 255 characters, and limited the **Number of values** to **1** because a job listing will only ever require one salary.

Next we add the **Closing date** field to which we assign the **Date** field type. The **Date** field type can be used for storing date ranges; it also features a popup date-picker calendar. When configuring this field we opt not to collect the **hour** and **minute** because it's very unlikely that a closing date will ever need that level of granularity. We do not collect an end date for the field because a closing is only ever a single date.

The next field we add is the **Related documents** field, which we add as a **File** field type. We add this field so that the editor can upload documents relating to the job. In the **Allowed file extensions** field we limit the files that can be uploaded here to only the common document formats for security reasons. To complete the **Related documents** field we set the **Number of values** as **Unlimited**; this allows the editor to upload as many files as necessary.

For the **Position type** field we apply the **List (text)** field type. This is a flexible type which allows data entry using a drop-down list, checkbox list, and radio buttons. In the **Allowed values list** we enter our values for the field on a separate line. Each line needs to contain two components, the machine-readable name for the option and the human readable name, separated by a pipe "|". We only allow one value for this field. If only one value is allowed for a **List** field, then the form will output a radio list. If the allowed values is greater than one, then the form will output a list of checkboxes, allowing multiple selections.

Finally, in the **MANAGE DISPLAY** configuration we rearrange the fields, and configure the labels to display inline to improve the display of the data.

See also

In Chapter 5, *Using Views to Create Custom Lists, Grids, and Tables*:

- ▶ *Creating a news listing view*

In Chapter 9, *Creating Regular, Mobile, and Tablet Themes*:

- ▶ *Overriding HTML output of a content type*

Building a custom content importer

Entering content is usually a simple task, but when there are batch loads of content to be inputted, then it becomes very tedious. In this recipe we will use the Drupal Feeds module to speed up this task by creating a batch content importer, which will process content in the form of a CSV file into content nodes on our site.

Getting ready

For this chapter you need to install and enable the following modules:

- ▶ Feeds (enable the Feeds and Feeds Admin UI features):
<http://drupal.org/project/feeds>
- ▶ Job scheduler: http://drupal.org/project/job_scheduler
- ▶ Chaos tools (if you haven't already installed it):
<http://drupal.org/project/ctools>

You will also need spreadsheet software such as Microsoft Excel, that is capable of saving files as CSVs.

How to do it...

We will be building a content importer for the Basic page content type. To do this we will first create a new importer. We will then configure the reporter so it knows how to process the import file, and finally we will download a template and import some content:

1. Select **Structure** from the admin menu, then select **Feed importers**.
2. Select the **+Add importer** tab.
3. In the **Name** field enter **Basic page importer**.
4. For the **Machine name** enter **basic_page_importer**.
5. In the **Description** field enter **Use this importer to import content from a CSV file into Basic page nodes**.
6. Click on **Create** to continue.
7. Under the **Basic settings** heading, select **Settings**.
8. In the **Attach to content type** field ensure that **Use standalone form** is selected in the drop-down.
9. In the **Periodic import** field select **Off**.
10. Ensure that **Import on submission** is checked and then click on **Save**.

Basic settings	Basic settings	Help
Attached to: [none] Settings Periodic import: off Import on submission	Name * <input type="text" value="Basic page importer"/> <small>A human readable name of this importer.</small>	
Fetcher Change	Description <input type="text" value="Use this importer to import content from a CSV file into Basic page"/> <small>A description of this importer.</small>	
File upload Settings <small>Upload content from a local file.</small>		
Parser Change	Attach to content type <input type="button" value="Use standalone form"/> <small>If "Use standalone form" is selected a source is imported by using a form under http://drupal7cbc1.streetfish.co.uk/import. If a content type is selected a source is imported by creating a node of that content type.</small>	
CSV parser Settings <small>Parse data in Comma Separated Value format.</small>		
Processor Change	Periodic import <input type="button" value="Off"/> <small>Choose how often a source should be imported periodically. Requires cron to be configured.</small>	
Node processor Settings <small>Create and update nodes.</small>	<input checked="" type="checkbox"/> Import on submission <small>Check if import should be started at the moment a standalone form or node form is submitted.</small>	
	<input type="checkbox"/> Process in background <small>For very large imports. If checked, import and delete tasks started from the web UI will be handled by a cron task in the background rather than by the browser. This does not affect periodic imports, they are handled by a cron task in any case. Requires cron to be configured.</small>	
	<input type="button" value="Save"/>	

11. Under the **Fetcher** heading, select **Change**.
12. Select the **File upload** option and click on **Save**.
13. Under the **Parser** heading, select **Change**.
14. Select the **CSV parser** option and click on **Save**.
15. Under the **Node processor** heading, select **Change**.
16. In the **Update existing nodes** field, select **Update existing nodes**.
17. For the **Content type** field, select **Basic page**:

Basic settings	Settings for Node processor	Help
Attached to: [none] Settings Periodic import: off Import on submission	Update existing nodes <input checked="" type="radio"/> Do not update existing nodes <input type="radio"/> Replace existing nodes <input type="radio"/> Update existing nodes (slower than replacing them) <small>Existing nodes will be determined using mappings that are a "unique target".</small>	
Fetcher Change File upload Settings <small>Upload content from a local file.</small>	Text format * <input type="button" value="Plain text"/> <small>Select the input format for the body field of the nodes to be created.</small>	
Parser Change CSV parser Settings <small>Parse data in Comma Separated Value format.</small>	Content type <input type="button" value="Basic page"/> <small>Select the content type for the nodes to be created. Note: Users with "import basic_page_importer feeds" permissions will be able to import nodes of the content type selected here regardless of the node level permissions. Further, users with "clear basic_page_importer permissions" will be able to delete imported nodes regardless of their node level permissions.</small>	
Processor Change Node processor Settings <small>Create and update nodes.</small> Mapping		

18. Click on **Save**.
19. Now select **Mapping**.
20. In the **Source field** enter **ID**, select the **Target** as **GUID**, then click on **Add**.
21. For the new **ID** row check the **Unique target** field, and click on **Save**.
22. In the **Source** field enter **Title**, select the **Target** as **Title**, then click on **Add**.
23. In the **Source** field enter **Body**, select the **Target** as **Body**, then click on **Add**.

Custom Content Types —————

24. Click on **Save**.

SOURCE	TARGET	UNIQUE TARGET	
ID	GUID	<input checked="" type="checkbox"/>	<input type="checkbox"/> Remove
Title	Title	<input type="checkbox"/>	<input type="checkbox"/> Remove
Body	Body	<input type="checkbox"/>	<input type="checkbox"/> Remove
Name of source field	Select a target <input type="button" value="▼"/>		<input type="button" value="Add"/>

25. Open the URL: `Drupal root/import`.
26. Select your new **Basic page importer**.
27. In the **Input** fieldset select **Download a template**.
28. After downloading the CSV file, open it with your spreadsheet software.
29. In row 1 enter **1** in the **ID** column.
30. In row 1 enter **Test imported content** in the **Title** column.
31. In row 1 enter **This is the body text of the page** in the **Body** column.
32. Save your file. It may ask you to confirm that you want to save as a CSV, in which case select **Yes**.
33. Go back to the `Drupal root/import` page in the browser.
34. Under the **File** heading, select **Choose file**.
35. Find and choose your newly created CSV file.
36. Click on **Import**.
37. You will now see your newly imported content in the content library.

How it works...

We start by naming the new importer. We add a human-readable name and a machine-readable name and also a description to help identify the importer should there be more than one at a later date.

We now begin to configure the importer. We first configure the importer to turn **Periodic import Off** because we want to import our content manually each time. However, it is possible to configure the importer to periodically import a content file from a predetermined URL.

When we come to the **Parser settings**, we set the importer to parse the content as CSV. This is the easiest format to use for importing content from a spreadsheet. However, it is also possible to import content from RSS feeds.

In the **Node processor** section, we select the option **Update existing content**; this means that all content imported will be created as a new node unless its **ID** matches an existing node, in which case it replaces the existing content. We then configure the **Content type**. This option causes the importer to treat imported content as a Basic page, and to create the new content with the Basic page content type.

The final part of the configuration is the **Mapping**. This is where we tell the importer which targets to map the CSV fields to. When the importer is running, it attempts to match each field in the CSV to each field in the Basic page node, according to the mapping setup in this section. So, if you happen to change your field name at a later date, then you would need to update the mapping configuration of the importer.

When we create the field mappings we first add an **ID** field. This is to uniquely identify the item of content so that if an item being imported matches an existing item with the same **ID** then that particular item will be updated, rather than a new node being created for it.

Finally we move on to testing out the importer. First we download a template that the system generates. We then use this to enter some sample data. Be sure that the **ID** column is unique to avoid overwriting nodes with the wrong content.

Building a forum

Most web developers will, at some point, need to implement a forum. Drupal makes this very easy to do as it contains a forum module in its core. In this recipe, we will create a new forum and then we will configure the settings for the forum.

Getting ready

For this recipe you need to ensure that you have enabled the Forum module, which is a Drupal core module.

How to do it...

To begin we are going to create a new forum; we will then add the forum to the main menu, and finally we will configure the Forum so that new posts cannot be added to any menus:

1. Select **Structure** from the admin menu, then select **Forums**.
2. Select **+Add forum**.
3. In **Forum name** enter **Support forum**.
4. In the **Description** field enter **Customer support topics**.

Custom Content Types

5. In the **Parent** field ensure that <root> is selected.
6. Click on **Save**.

The screenshot shows a configuration form for a new forum. The fields are as follows:

- Forum name ***: Support forum. A descriptive note below says: "Short but meaningful name for this collection of threaded discussions."
- Description**: Customer support topics. A large text area with a placeholder: "Description and guidelines for discussions within this forum."
- Parent ***: <root>. A dropdown menu with a note: "Forums may be placed at the top (root) level, or inside another container or forum."
- Weight**: 0. A dropdown menu with a note: "Forums are displayed in ascending order by weight (forums with equal weights are displayed alphabetically)."

7. Select the link to your new **Support forum** to open the forum.
8. Copy the link to your forum, for example, `forum/7`.
9. Select **Structure** from the admin menu, then select **Menus**.
10. In the **Main menu** row select **add link**.
11. In the **Menu link title** enter **Support forum**.
12. In the **Path** field paste the copied link to the support forum.
13. Ensure that <Main menu> is selected for the **Parent link** field.
14. Click on **Save**.
15. Select **Structure** from the admin menu and then select **Content types**.
16. In the **Forum topic** row click on **edit**.
17. Select the **Menu settings** tab.
18. Uncheck the **Main menu** checkbox.
19. Click on **Save content type**.

How it works...

When creating a new forum we are given two options: Add container and Add forum. If you are planning on having multiple forums, then it may be useful to group similar forums into a forum container. In this recipe we do not need that level of categorization, so we simply create a forum without a container.

After we have created the forum, we add it to the main menu. We do this by manually copying the link to the forum and inserting it into the path field.

Finally, we configure the settings of the **Forum topic** content type. We uncheck the **Main** menu checkbox in the **Menu settings**. This means that when the user creates a new forum topic, they won't have the option to add the post to a menu.

There's more...

As you have seen, it is relatively quick to set up a new forum that will satisfy most users' needs. However, there may be cases that demand more functionality from your forum, such as providing more fields for your forum post content type and limiting access to forum posts.

Forum access

Many websites require a private members area with private members only forums. This is not possible without a third-party module. This functionality can be implemented using the Forum access module, which allows you to give view, post, edit, and delete permissions to each user role.

Extending the forum content type

Forum topics are like regular content types in that it is possible to add any number of extra fields. For example, you could provide an image field for users to upload a screenshot with their post.

See also

- ▶ Forum access module: http://drupal.org/project/forum_access

5

Using Views to Create Custom Lists, Grids, and Tables

In this chapter we will cover:

- ▶ Creating a news listing view
- ▶ Creating a dynamic view
- ▶ Creating a latest news block
- ▶ Building a news image grid view
- ▶ Creating a randomly selected list of images
- ▶ Creating an archived content block and view
- ▶ Building complex views using relationships
- ▶ Using attachments to extend views' filtering functionality
- ▶ Using attachments to extend views' output

Introduction

Throughout this chapter we will explore some of the most common features of the Views module, from the basic to the advanced. Views makes it easy to create listings of your content, such as job listings or a list of products in a shop. It also allows you to set filters, and to expose those filters to the user. For example, in a listing of products in a shop, you could output a price filter, allowing users to select a price range.

Using Views to Create Custom Lists, Grids, and Tables

Views is the most popular module on [Drupal.org](#) due to its power and flexibility, and by the end of this chapter, I'm sure you'll see why!

We begin by creating a simple news listing view, which we then expand upon in the following chapter by making it dynamic. We then see how the Views module can be used to create custom blocks by creating a Latest news block. We then create a view with a grid view format, which demonstrates how we can configure a view to display content formatted into grids of images; we also look at how content can be outputted with a random sorting order. Next, we create an archived news view that builds upon the previously created dynamic view; we also create an archived view block which outputs a list of months to filter the view on.

We then move on to the more complex task of using relationships in a view, which enables us to include more fields in the output. Finally, we explore the functionality of the Attachment display type, which enables us to combine displays in a view to create compound outputs.

Creating a news listing view

In this recipe we will be creating a new content type, **News article**, which will be used throughout the chapter. We will then use the Views module to create a News view to display a News article listing. The expected result will look as follows:

Sample news article

 Submitted by [dylan](#) on Fri, 12/23/2011 - 15:06



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis lectus metus, at posuere neque. Sed pharetra nibh eget orci convallis at posuere leo convallis. Sed blandit augue vitae augue scelerisque bibendum. Vivamus sit amet libero turpis, non venenatis urna. In blandit, odio convallis suscipit venenatis, ante ipsum cursus augue.

[Read more](#)

Sample news content 2

 Submitted by [dylan](#) on Tue, 12/13/2011 - 21:48



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis lectus metus, at posuere neque. Sed pharetra nibh eget orci convallis at posuere leo convallis. Sed blandit augue vitae augue scelerisque bibendum. Vivamus sit amet libero turpis, non venenatis urna. In blandit, odio convallis suscipit venenatis, ante ipsum cursus augue.

[Read more](#)

Getting ready

For this recipe you will need to install and enable the most recent recommended releases of the following modules:

- ▶ <http://drupal.org/project/views>
- ▶ <http://drupal.org/project/ctools>

How to do it...

To begin we will create a News article content type to which we will add an **Image** field. After configuring the **Image** field we will then create a new View that we will configure to output a list of teasers of the News article type. Finally, we will configure the View to add a **News menu** item to the **Main menu**.

1. Select **Structure** from the admin menu, then select **Content types**.
2. Select **+Add content type**.
3. Enter **News article** in the **Name** field.
4. In the **Description** field, enter **Use the News article content type to create news articles**.
5. Click on **Save and add fields**.
6. Under **Add new field** enter **Image**.
7. In the **NAME** column enter **news_image**.
8. Select **Image** in the **FIELD** drop-down.

LABEL	NAME	FIELD	WIDGET	OPERATIONS
⊕ Title	title	Node module element		
⊕ Body	body	Long text and summary	Text area with a summary	edit delete
⊕ Add new field				
	<input type="text" value="Image"/> <input type="text" value="field_news_image"/> <input type="button" value="Image"/> <input type="button" value="Image"/> Label Field name (a-z, 0-9, _)	Type of data to store.		Form element to edit the data.
⊕ Add existing field				
	<input type="text"/> <input type="button" value="- Select an existing field -"/> Label Field to share		<input type="button" value="- Select a widget -"/> Form element to edit the data.	

9. Click on **Save**.
10. On the **Field settings** page click on **Save field settings**.
11. On the **News article settings** popup, leave all settings in their default states and click on **Save settings**.
12. Select the **Manage display** tab, then select the **Teaser** sub-tab.

13. Select the gear icon for the **Image** field:

FIELD	LABEL	FORMAT	
+ Image	<Hidden>	Image	Image style: thumbnail 
+ Body	<Hidden>	Summary or trimmed	Trim length: 600 

14. From the **Image style** drop-down, select **Medium**, then click on **Update**.

15. Select **<Hidden>** from the **Image** row's **LABEL** drop-down.

16. Click on **Save**.

17. Select **Structure** from the admin menu, then select **Views**.

18. Select **+Add new view**.

19. Enter **News** in the **View name** field.

20. Ensure that the first fieldset reads as follows: Show **Content** of type **News article** sorted by **Newest first**.

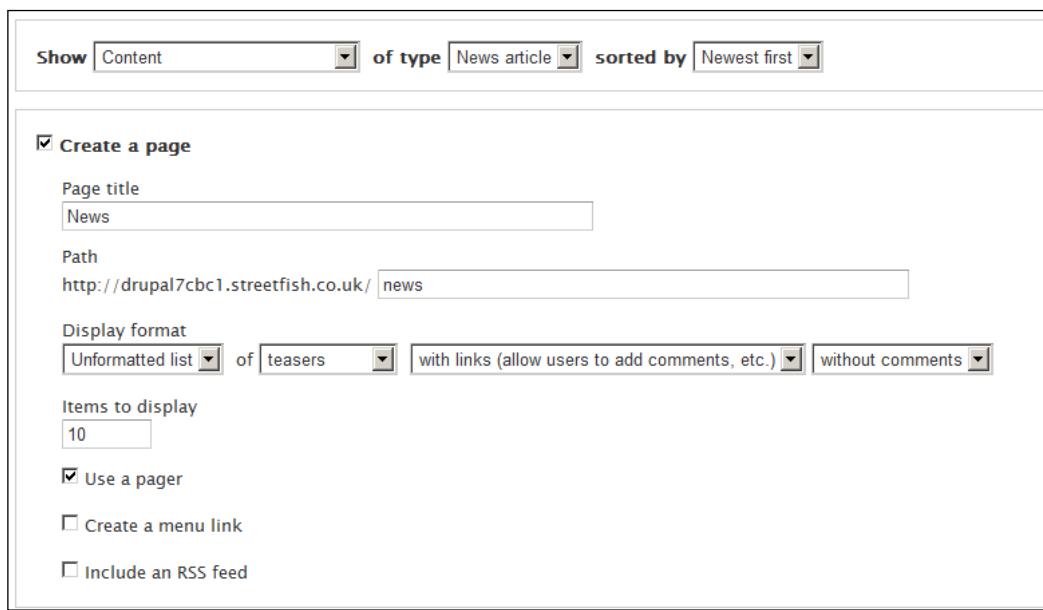
21. In the second fieldset, leave the **Page title** and **Path** set as **News**.

22. In the **Display format** section, configure the drop-downs so they read, **Unformatted list of Teasers with links (allow users to add comments, etc.) without comments**.

23. Leave the **Items to display** field set as **10**.

24. Ensure that **Use a pager** is checked.

25. Click on **Continue and edit**.



The screenshot shows the 'Create a page' configuration form for the 'News' view. At the top, there are dropdown menus for 'Show' (Content), 'of type' (News article), and 'sorted by' (Newest first). Below these are several configuration options:

- Create a page** checkbox (checked):
Page title: News
Path: http://drupal7cbc1.streetfish.co.uk/ news
- Display format**: Unformatted list of teasers with links (allow users to add comments, etc.) without comments
- Items to display**: 10
- Use a pager** checkbox (checked)
- Create a menu link** checkbox (unchecked)
- Include an RSS feed** checkbox (unchecked)

26. Under the **Page settings** section, select the link next to **Menu**, which reads **No menu**:

The screenshot shows the 'Page settings' section of a Views configuration page. The 'Menu' field is circled in red, containing the text 'No menu'. Other visible fields include 'Path: /news', 'Access: Permission', and various header, footer, and pager settings.

27. In the popup window, select **Normal menu entry** under the **Type** field.
28. In the **Title** field, enter **News**.
29. In the **Description** field, enter **News listings page**.
30. Select **Main menu** for the **Menu** field.
31. Click on **Apply**, this will take you back to the Views edit page.
32. Click on **Save** to save the view's configuration.
33. You will now see a news link in your main menu; add some News article nodes to test the new view.

How it works...

We begin by creating a news content type that will be used throughout this chapter. It's a simple content type similar to Article.

After creating the News content type we create a new View called News. In the view setup we specify that we want to show **Content** of type **News article** sorted by **Newest first**. This is to say that we want to show content nodes, of the News article type, with a sorting order of newest first. We then configure the **Display** format so that it reads **Unformatted list of Teasers with links (allow users to add comments, etc.) without comments**. This is to say that we want to display the content in an unformatted form, and to display each item in the **Teaser** view mode, which only shows an excerpt of body text, 600 characters by default.

On the view's configuration page, the only change we make is to the **Menu** settings. We configure the view so that it adds a menu item to the main menu. Alternatively, this could be done manually on the **Menus** configuration page.

There's more...

In this recipe we have seen how to create a simple view, which we have added directly to the main menu. However, there are other ways to add views to a site, and there are also ways to configure a view to achieve a more bespoke output.

Adding views using Panels

The Panels module allows you to create custom pages. On a page, it is possible to create custom layouts to which you can add existing content nodes, views, and many other features. You may, for example, want to display two views on a page, with a basic page node at the top of the page, and a custom view underneath. Panels makes this kind of customization very easy to do.

Alternative listings layouts

This recipe shows how to create a news listings view that displays a list of 10 article teasers. It is likely that you will want to display a thumbnail to accompany the news items. You can of course configure the teaser view mode to display the News article's image with the listings, but without applying custom CSS code, the format may not be as you would expect.

An alternative to displaying items using the Teaser configuration would be to display as a table. By displaying as a table you can configure the view so that the image displays in column 1, and the title and body display in column 2, separated by a `
` tag.

See also

In this chapter:

- ▶ *Using attachments to extend Views' output*

Creating a dynamic view

In the previous recipe we saw how to create a view with a static filter applied to it. In this recipe we will be creating a view where the filter can be dynamically set by the calling URL, which contains the arguments. We will also be setting up a view which is generated from a URL in the following format:

`http://www.example.com/news/8`

This URL contains the argument "8", which will be used to filter the view to display only news stories from category 8, as in the following screenshot:

Sample news content 2

 Submitted by [dylan](#) on Tue, 12/13/2011 - 21:48



Placeholder text for news item 2: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis lectus metus, at posuere neque. Sed pharetra nibh eget orci convallis at posuere leo convallis. Sed blandit augue vitae augue scelerisque bibendum. Vivamus sit amet libero turpis, non venenatis urna. In blandit, odio convallis suscipit venenatis, ante ipsum cursus augue.

[Read more](#)

News item 1

 Submitted by [dylan](#) on Sat, 12/10/2011 - 12:15



Placeholder text for news item 1: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis lectus metus, at posuere neque. Sed pharetra nibh eget orci convallis at posuere leo convallis. Sed blandit augue vitae augue scelerisque bibendum. Vivamus sit amet libero turpis, non venenatis urna. In blandit, odio convallis suscipit venenatis, ante ipsum cursus augue.

This technique can help to increase the flexibility of your view and to dramatically reduce the build time of your site when you are using a view in multiple locations.

Getting ready

To complete this recipe you will need to have first completed the preceding recipe, *Creating a news listing view*. We will be building upon both the News article content type and the News view.

How to do it...

We will begin by creating a new Categories taxonomy for categorizing News articles. We will then update the News article content type to allow articles to be categorized using terms from the Categories taxonomy. Following this we will update the News view to include a dynamic filter, which filters items using a variable from the URL. Finally, we will add the filtered view to the **Main menu** so that we can test it.

1. Select **Structure** from the admin menu, then select **Taxonomy**.
2. Select **+Add vocabulary**.
3. Enter **Categories** in the **Name** field.
4. Leave the **Description** field blank and click on **Save**.
5. On the **Taxonomy** list page select **add terms** for the **Categories** row.
6. In the **Name** field enter **Category 1**, then select **Save**.
7. In the **Name** field enter **Category 2**, then select **Save**.
8. In the **Name** field enter **Category 3**, then select **Save**.
9. Make a note of the ID for **Category 1** by hovering over the **edit** link, and note the number immediately following the `/term/` component of the path; for example, in the case of the path `/term/8` note down the number 8.
10. Select **Structure** from the admin menu, then select **Content types**.
11. Select **Manage fields** for the **News article** content type.
12. In the **Add new field** section enter **Category** as the **Label**.
13. Enter **news_category** in the **NAME** column.
14. Select **Term reference** in the **FIELD** drop-down.
15. Select **Check boxes/radio buttons** for the **WIDGET** field.

LABEL	NAME	FIELD	WIDGET	OPERATIONS
⊕ Title	title	Node module element		
⊕ Body	body	Long text and summary	Text area with a summary	edit delete
⊕ Image	field_news_image	Image	Image	edit delete
⊕ Add new field	Category <input type="text" value="field_news_category"/> Label <input type="text" value="Field name (a-z, 0-9,_)"/>	Term reference <input type="button" value="▼"/> Type of data to store.	Check boxes/radio buttons <input type="button" value="▼"/> Form element to edit the data.	
⊕ Add existing field	<input type="text"/> Label <input type="text" value="Field to share"/>	- Select an existing field - <input type="button" value="▼"/> Field to share	- Select a widget - <input type="button" value="▼"/> Form element to edit the data.	

16. Click on **Save**.
17. On the **Field settings** page select **Categories** from the **Vocabulary** drop-down.
18. Click on **Save field settings**.
19. On the **News article settings** page, leave the values in their default states and click on **Save settings**.
20. Select **Structure** from the admin menu, then select **Views**.
21. Select **Edit** on the **News view**.
22. Expand the **Advanced** fieldset, then select the **add** button next to **CONTEXTUAL FILTERS**:

The screenshot shows the 'Edit view name/description' dialog for a 'Page' view. The 'Advanced' fieldset is expanded, revealing the 'CONTEXTUAL FILTERS' section. The 'add' button in this section is circled in red, indicating it as the target for step 22. Other sections visible include 'PAGE SETTINGS', 'RELATIONSHIPS', 'NO RESULTS BEHAVIOR', 'EXPOSED FORM', and 'OTHER'.

23. In the **Add contextual filter** popup enter **category** in the **Search** field.

24. Check the option: **Content: Category (field_news_category)**:

Add contextual filters

Search Filter

Content: Category (field_news_category)
Appears in: node:news_article.

25. Select **Add and configure contextual filters**.

26. In the **Configure contextual filters** popup ensure that **Display all results for the specified field** is selected.

27. In the secondfieldset, check the **Specify validation criteria** option.

28. Select **Taxonomy term** in the **Validator** field.

29. Check the **Categories** option.

30. Select the **Term ID** option from the **Filter value type** field.

31. Select **Display all results for the specified field** for the **Action to take if filter value does not validate** field.

WHEN THE FILTER VALUE IS IN THE URL OR A DEFAULT IS PROVIDED

Override title
 Override breadcrumb
 Specify validation criteria

Validator

Vocabularies
 Forums
 Categories
 Tags

If you wish to validate for specific vocabularies, check them; if none are checked, all terms will pass.

Filter value type

Select the form of this filter value; if using term name, it is generally more efficient to convert it to a term ID and use "Taxonomy: Term ID rather than Taxonomy: Term Name" as the filter.

Transform dashes in URL to spaces in term name filter values

Action to take if filter value does not validate

32. Select **Apply (all displays)**.
33. On the **News view's edit** page, click on **Save**.
34. Select **Structure** from the admin menu, then select **Menus**.
35. Select **add link** for the **Main menu** row.
36. In the **Menu link title** field, enter **Category 1 news**.
37. For the **Path**, enter `news/` followed by the category ID we made a note of at step 9, for example, `news/8`.
38. Ensure that **<Main menu>** is selected for **Parent link** and then click on **Save**.
39. You will now see a news page that is filtered to only show news items from Category 1, where the filter argument is taken from the URL. To test the filtered view, add some News article nodes with the **Category** set to **Category 1**.

How it works...

We begin this recipe by creating a new vocabulary. Vocabularies are sets of terms that can be used for many purposes. In this case, we are creating a vocabulary to store a set of categories with which we can associate news items so that news items can be filtered to only show a particular category of news.

After creating the **Category taxonomy** we update the existing News article content type so that it has a **Category** field. The field type is set to **Term reference**. The **Term reference** field allows us to create drop-down lists, radio buttons, or checkboxes for the field, where the list of possible values is taken from a predefined taxonomy, in this case the **Categories taxonomy**. We set the widget type as **Check boxes/radio buttons**, and on the **Field settings** page, we set the **Number of values** to **1**, which means that the field will be outputted as a set of radio buttons. If we chose any number greater than one, then the field would be outputted as a set of checkboxes.

Now that the News content type has been updated, we proceed to update the News view. We add a **Contextual filter** to the view, this is like a regular filter, but rather than filtering on a static value, the filter's value can be passed in as an argument. The argument can be passed in by various means, but in this recipe we will be passing the argument in through the URL of the view. After adding our **Category filter**, we continue to the configure contextual filters popup, where first we configure the action we would like if there is no argument passed in. We select **Display all results for the specified field**, which will output an unfiltered view, when there are no arguments in the URL. We then specify a validation criterion for when there *is* an argument passed in. We choose to validate the value against a Taxonomy term, specifically the Categories taxonomy. This means that if the filter value is not an ID number of a Taxonomy term from the Categories taxonomy, then the view will display all results.

Finally, we add a new menu item to the **Main menu** to display our new filtered news view. To do this we enter the path to the News view, followed by the ID corresponding to the **Taxonomy category** we want to filter.

There's more...

We have seen how we can build a dynamic view by passing arguments into the view's URL, but there are other ways to do this which can be much more useful.

Passing arguments to a view from a Panel

Imagine you had three different news categories on a site, and you wanted to create three blocks to show news teasers from each of these three categories. You could create three different displays in a News view, one for each category, each with a static filter set to the relevant category. However, a more efficient method would be to create one display in the view, and to add a dynamic filter on the Category (much like we have done in this recipe). Then, instead of adding the view directly to a menu, you could attach three instances of the view to a Panel page, in each case setting a different argument for the Panel to pass to the view.

Using this method you will only need to manage one display in the News view, which reduces the build time and keeps errors to a minimum.

Creating a latest news block

One of the useful features of Views is that it allows us to create custom blocks that can be used to display our content in many different ways. In this recipe we will be using the News article content type and expanding on the News view to create a block that we will use to display the five most recent News articles added to the site.

Getting ready

To complete this recipe you will need to have first completed the first recipe of this chapter, *Creating a news listing view*. We will be building upon both the News article content type and the News view.

How to do it...

To complete this recipe we are first going to modify the existing News view by adding a block display to it. We will then configure the block display to display a list of five titles, and finally we will configure the new block to be displayed in the Sidebar first region of the page.

1. Select **Structure** from the admin menu, then select **Views**.
2. Select **edit** for the **News view**.

3. Select the **+Add** button on the display bar, then select **Block**:



4. In the **Format** section select **Content**, then select the **Fields** option.
5. Change the **For** drop-down to **This block (override)**:

Block: How should each row in this view be styled

For This block (override) ▾

Content
 Fields

You may also adjust the [settings](#) for the currently selected row style.

6. Select **Apply (this display)**.
7. On the following options page, leave the options in their default state and select **Apply (this display)**.
8. In the **BLOCK SETTINGS** section, select the link that reads **None** next to **Block name**.
9. In the popup, enter **Latest news** in the textfield and click on **Apply**.
10. In the **PAGER** section select the link that reads **Full** next to the **Use pager** label.
11. In the **For** drop-down, select **This block (override)**.
12. Select **Display a specified number of items**, then select **Apply (this display)**.
13. In the following pager options popup change **Items to display** to **5**.

14. Select **Apply (this display)**.

15. Click on **Save** to save the view.
16. Select **Structure** from the admin menu, then select **Blocks**.
17. Look for the block named **Latest news**, and under its region drop-down select **Sidebar first** (or another suitable region if not using the Bartik theme).
18. Click on **Save blocks**.
19. Visit the home page to see the block displaying five of the latest news item titles.

How it works...

We begin by adding a new **Block** display to the News view. This creates a new display in the News view. Note that most of the settings in the **Block** are inherited from the **Page** display. We can then override the settings that we want to be different in the block display. Firstly we change the **Format** of the view so that we output **Fields** rather than a list of Teasers. This provides a vastly increased level of control over the output. After selecting that we want to display **Fields**, the block display automatically adds the **Title** field to the list of fields to display, and for this recipe, that is the only field we will need.

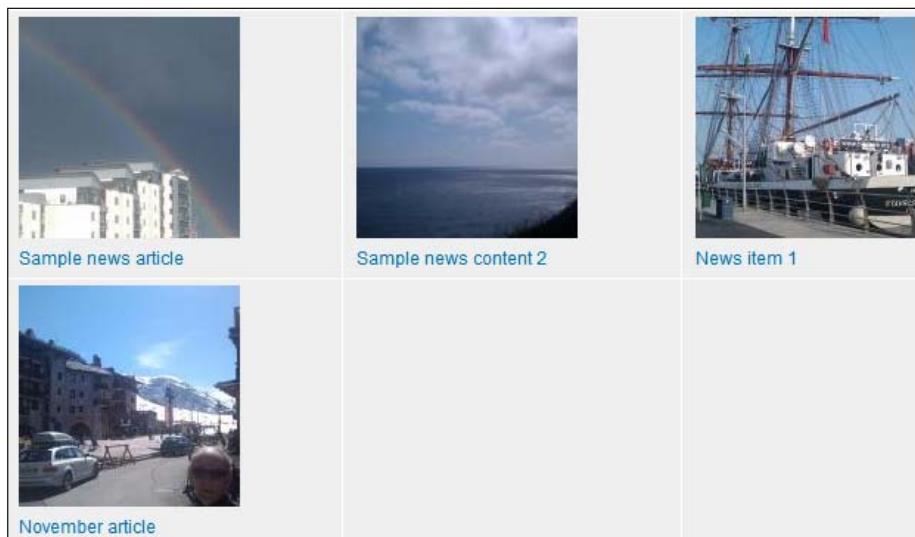
When changing the settings in an overridden display such as our new block display, it's important to set the **For** drop-down to **This block (override)**; otherwise, any changes you make will be applied to every display. Overridden fields are italicised so you can easily see which fields have been altered in the current view.

After setting the format of the new display we set the **Block name**, which will be the name that is listed on the **Blocks listing** page. Following this we move on to the **Pager settings**, where we remove the paging links from the block and set it to display only five links.

Finally, we open the **Blocks page** and set the new Latest news block to display in the Sidebar first region of the theme.

Creating a news image grid view

Using the Drupal Views module it is possible to create a variety of different display formats including tables, lists, and grids. In this recipe, we will create a grid view to display thumbnail images and title links for the most recent News articles, as shown in the following screenshot:



Getting ready

To complete this recipe you will need to have first completed the first recipe of this chapter, *Creating a news listing view*. We will be making use of the News article content type and also adding a new display to the News view.

How to do it...

We will begin this recipe by creating a new page display in the News view, which we will then configure to display a grid of images and titles to represent the latest 12 News articles:

1. Select **Structure** from the admin menu, then select **Views**.
2. Select **edit** for the **News** view.

3. Select the **+Add** button on the display bar, then select **Page**:



4. In the **Format** section select **Unformatted list**.
5. Change the **For** field to **This page (override)**.
6. Select the **Grid** option, then select **Apply (this display)**.
7. On the **Style options** popup, change the **Number of columns** field to **3**.
8. Select **Apply (this display)**.
9. In the **Format** section select **Content**, then select the **Fields** option.
10. Change the **For** drop-down to **This block (override)**.
11. Select **Apply (this display)**.
12. On the following options page, leave the options in their default state and select **Apply (this display)**.
13. In the **Fields** section select **Add**.
14. In the **Search** field enter **image**.
15. Check the **Content: Image – Appears in: node:news_article** option:

A screenshot of a 'Add fields' dialog. At the top left is the title 'Add fields'. Below it is a search bar containing 'image' and a 'Filter' dropdown set to '- All -'. There are two options listed:

- Content: Image
Appears in: node:article.
- Content: Image
Appears in: node:news_article.

Below the list is a summary section titled 'Selected: Content: Image'. At the bottom are two buttons: 'Add and configure fields' and 'Cancel'.

16. Click on **Apply (all displays)**.
17. On the **Configure field** popup select **This page (override)** in the **For** field.
18. Uncheck the **Create a label** field.
19. For the **Image style** field, select **thumbnail**.
20. For the **Link image to** field, select **Content**.

Configure field: Content: Image

For This page (override) ▾

Appears in: node:news_article.

Create a label
Enable to create a label for this field.

Exclude from display
Enable to load this field as hidden. Often used to group fields, or to use as token in another field.

Formatter
Image ▾

Image style
thumbnail ▾

Link image to
Content ▾

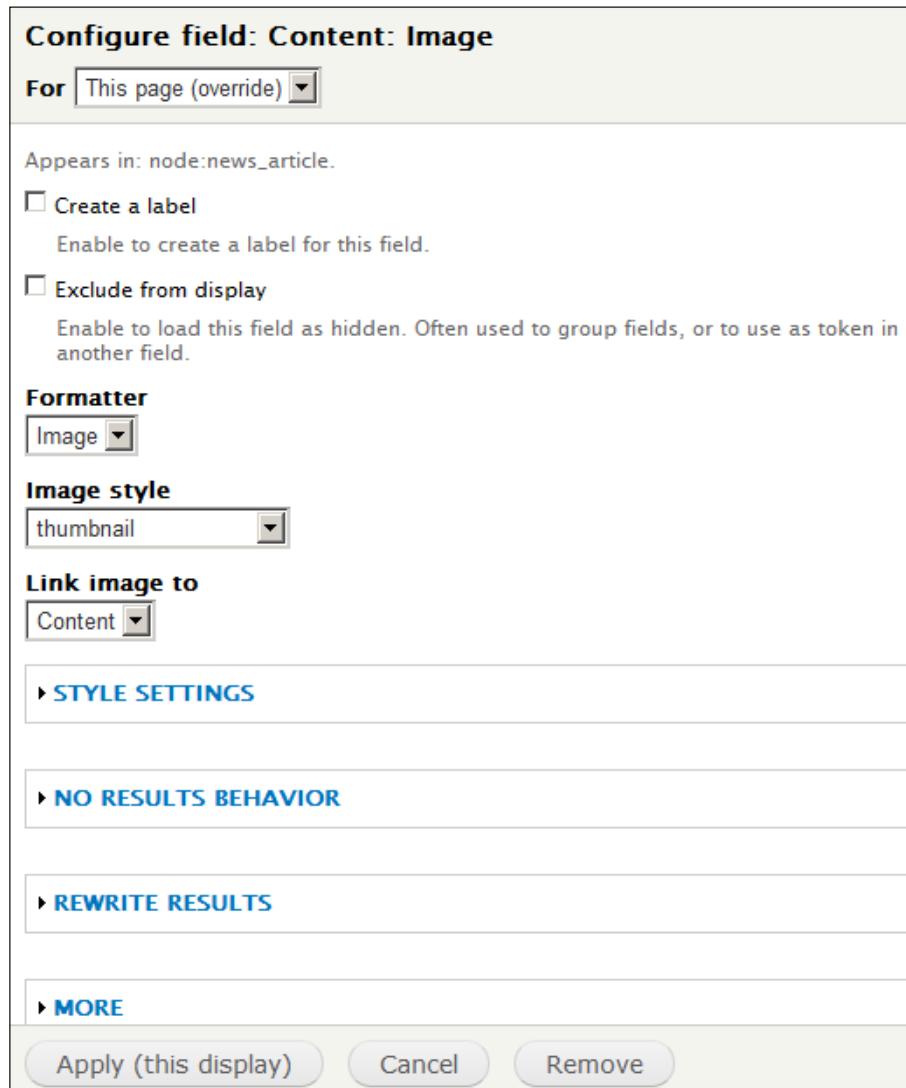
▶ **STYLE SETTINGS**

▶ **NO RESULTS BEHAVIOR**

▶ **REWRITE RESULTS**

▶ **MORE**

Apply (this display) **Cancel** **Remove**



21. Select **Apply (this display)**.
22. In the **Fields** section, select the drop-down on the **add** button, then select **rearrange**.

23. Using the drag handles, drag the **Content:image** field above the **Content:title** field.
24. Select **Apply (this display)**.
25. In the **Page settings** section select the link in the **Path** field.
26. Enter news/grid-view and select **Apply**.
27. In the **Pager section**, select the option **Paged, 10 items**.
28. Change the **For** field to **This page (override)**.
29. Change the **Items per page** field value from **10** to **12**.
30. Select **Apply (this display)**.
31. Click on **Save** to save the changes to the view.
32. Go to Drupal root/news/grid-view to see your new grid view; you may need to add some sample **News articles (with images)** to fully demonstrate the view.

How it works...

We begin by editing the News view to add a new page display. We then proceed to override the display by first changing the format from Unformatted list, to Grid, and then setting the number of columns to 3. Following this, again in the Format section, we change the Content output to Fields. This allows a greater level of flexibility in which data fields are outputted in the display.

After setting the display to output fields, we can see that the **Title** field is automatically added to the list of fields to output. We then add another field to output the image; it's important to check that the **Image** field is the correct one for the content type because other content types may have their own **Image** fields. We then configure the **Image** field by first removing the field label. We then set the **Image format** of the image so that it will output **Thumbnail** size. To complete the **Image** field configuration, we set the image to link to the content node that it relates to.

After we have added the image field, we have all the fields that are needed for the view. However, the order of output is set as follows:

1. Title
2. Image

For our Grid, we want to display the image above the title, so we then open the **Rearrange** popup for the **Fields**. Using the drag handles, we move the **Title** field below the **Image** field.

Finally, we set the path to the view as news/grid-view; we could have just entered the path as grid-view, but adding news views under a /news URL will help to group the views together and to reduce potential namespace conflicts.

Creating a randomly selected list of images

Space is always of a premium on the homepage of a site. One solution is to display a limited selection of random items of content. In this recipe, we will see how to create a display in a view where the outputted items are randomly selected from the pool of News article nodes.

Getting ready

To complete this recipe you will need to have first completed the first recipe of this chapter, *Creating a news listing view*. We will be making use of the News article content type and also adding a new display to the News view.

How to do it...

In the first part of this recipe we will be editing the News view to add a new display. We will then configure the new display to output three randomly chosen News article images.

1. Select **Structure** from the admin menu, then select **Views**.
2. Select **edit** for the **News** view.
3. Select **+Add**, then select **Block**:



4. Select the link next to **Display name**, and enter **Random news image** in the **Name** field.
5. Select **Content** in the **Format** section and select **This block (override)** in the **For** field.
6. Select the **Fields** option.
7. Select **Apply (this display)**.
8. Select the **Title** field.
9. Select **This block (override)** from the **For** drop-down.
10. Select **Remove**
11. Select the Image field, **Content: Image (Image)**.
12. Uncheck the **Create a label** field.
13. In the **Image style** drop-down, select **Thumbnail**.

14. Select **Apply (this display)**.
15. In the **Sort Criteria** section select **add**.
16. In the **Search** field, enter the search term, **random**.
17. Check the option **Global: Random** and select **Add and configure sort criteria**.
18. On the **Configure sort criterion** popup, select **This block (override)** in the **For** field, then select **Apply (this display)**.
19. Still in the **Sort Criteria** section, select the drop-down arrow next to the **add** button, then select the **rearrange** option.
20. In the **Rearrange sort criteria** popup, select the **Remove** link for the **Content: post date desc** row, so that **Global: Random asc** is the only option.
21. Select **Apply (this display)**.
22. In the **Pager** section select **Full** and set the **For** field to **This block (override)**.
23. Select the option **Display a specified number of items** and then select **Apply (this display)**.
24. In the **Pager options** popup set the **Items to display** to **3**, then select **Apply (this display)**.

The screenshot shows the configuration interface for a 'Random news image' view. The top navigation bar includes tabs for 'Page', 'Block', and 'Page', with the 'Page' tab selected. A search bar contains the text 'Random news image'. A '+ Add' button is visible, and an 'edit view name/description' dropdown is open. Below the tabs, the title 'Random News Image details' is displayed. The configuration is organized into several sections:

- TITLE:** Title: News
- FORMAT:** Format: Unformatted list | Settings. Show: Fields | Settings.
- FIELDS:** Content: Image
- FILTER CRITERIA:** Content: Published (Yes) | Content: Type (= News article)
- SORT CRITERIA:** Global: Random (asc)
- BLOCK SETTINGS:** Block name: None | Access: Permission | View published content.
- HEADER:** add
- FOOTER:** add
- PAGER:** Use pager: Display a specified number of items / 3 items | More link: No
- ADVANCED:** (Collapsible section)
 - CONTEXTUAL FILTERS:** add | Content: Category
 - RELATIONSHIPS:** add
 - NO RESULTS BEHAVIOR:** add
 - EXPOSED FORM:** Exposed form style: Basic | Settings
 - OTHER:** Machine Name: block_2 | Comment: No comment | Use AJAX: No | Hide attachments in summary: No | Use aggregation: No | Query settings: Settings | Field Language: Current user's language | Caching: None | Link display: Page | CSS class: None | Theme: Information | Block caching: Do not cache

25. Click on **Save**.
26. Select **Structure** from the admin menu, then select **Blocks**.
27. Find the new block **View: News: Random news image** and select **Configure**.
28. In the **Block title** field enter **<none>**.
29. In the **Region settings** section, set the **Bartik** drop-down to **Sidebar first**.
30. Click on **Save block**.
31. You will now see the random news image block in the left column. You may need to add some News article nodes, with images, to fully test this block.

How it works...

We begin this recipe by adding a new block display to the News view. We then configure the new display's format settings to output fields. We then remove the **Title** field from the display, and then configure the **Image** field to remove its label, and set it to output images in the thumbnail format.

We then begin to configure the **Sort criteria** for the display. We want to specify that the nodes outputted in the view are to be randomly selected from all of the News article nodes that are available. To do this we add the **Random** sort criterion. However, the **Post date** criterion still remains active, so we then remove this.

The final configuration we make to the view's display is to set the **Pager** settings to output three nodes without a pager.

To complete the recipe we assign our new block to the **Sidebar first** region of the theme.

Creating an archived content block and view

We have already seen how to create a News listing view. In this recipe we will build on the idea of the listing view and add a block which summarizes the News article content into a list of months and years, where the links outputted in the block open a view which is dynamically filtered to display articles from a specific month and year:

A screenshot of a 'News archive' block. The block has a light gray background and a thin black border. At the top, the text 'News archive' is centered in a dark font. Below this is a horizontal line. Underneath the line, there is a bulleted list of months and years, each followed by a small number in parentheses indicating the count of articles. The items in the list are: 'November 2011 (1)' and 'December 2011 (2)'. The text is in a dark font, and the numbers in parentheses are in a lighter blue color.

- November 2011 (1)
- December 2011 (2)

Getting ready

To complete this recipe you will need to have first completed the first recipe of this chapter, *Creating a news listing view*. We will be making use of the News article content type.

How to do it...

We will begin by creating a new View that will include Page and Block displays. We will then configure the settings of these displays by adding a dynamic filter, which filters on the Created date. Finally we will add the new block to the theme.

1. Select **Structure** from the admin menu, then select **Views**.
2. Select **+Add new view**.
3. In the **View name** field, enter **Archived news**.
4. In the firstfieldset, configure the drop-down menus to read: **Show Content of type News article sorted by Newest first**.
5. In the **Create a page** section, leave all options in their default states.
6. Check the field **Create a block**.
7. Set the **Items per display** field to **100**.

The screenshot shows the 'Create a page' configuration screen for a View named 'Archived news'. At the top, there are dropdown menus for 'Show' (set to 'Content'), 'of type' (set to 'News article'), and 'sorted by' (set to 'Newest first'). Below these are several configuration sections:

- Create a page**:
 - Page title**: Archived news
 - Path**: http://drupal7cbc1.streetfish.co.uk/archived-news
 - Display format**: Unformatted list of teasers with links (allow users to add comments, etc.) without comments
 - Items to display**: 10
 - Use a pager**: checked
 - Create a menu link**: unchecked
 - Include an RSS feed**: unchecked
- Create a block**:
 - Block title**: Archived news
 - Display format**: Unformatted list of titles (linked)
 - Items per page**: 100
 - Use a pager**: unchecked

8. Click on **Continue & edit**.
9. In the **Contextual filters** section, select **add**.
10. Enter the search term **created year** and select the option: **Content: Created year + month**:



11. Select **Select and configure contextual filters**.
12. In the **Configure contextual popup**, select **This page (override)** in the **For** field.
13. Leave all other options in their default state and select **Apply (this display)**.
14. Click on **Save**.
15. Select the **+Add** button, then select **Block**.
16. Select the **Block** display.
17. In the **Contextual filters** section, select the existing filter **Content: Created year + month**.
18. Select **This block (override)** from the **For** menu.
19. Under the heading **When the filter is not available**, select the **Display a summary option**.
20. Leave the **Sort order** as **Ascending**, **Sort by** as **Date**, and **Format** as **List**.

21. Enter **archived-news** in the **Base path** field:

Configure contextual filter: Content: Created year + month

For This block (override) ▾

WHEN THE FILTER VALUE IS NOT AVAILABLE

Display all results for the specified field
 Provide default value
 Hide view
 Display a summary

Sort order
 Ascending
 Descending

Sort by
 Date
 Number of records

Format
 List
 Unformatted
 Jump menu

Base path
archived-news

Define the base path for links in this summary view, i.e.
http://example.com/your_view_path/archive.
Do not include beginning and ending forward slash. If this value is empty, views will use the first path found as the base path, in page displays, or / if no path could be found.

22. Select **Apply (this display)**.
23. Click on **Save**.
24. Select **Structure** from the admin menu, then select **Blocks**.
25. Find the newly created block **View: Archived news** and its **configure** option.
26. In the **Block title** field enter **News archive**.
27. In the **Region settings** select **Sidebar first** for the **Bartik** field (or select another suitable region if not using the Bartik theme).

28. In the **Show block on specific pages** section, select the option: **Only the listed pages** and enter the text **news-archive*** in the field below.
29. Click on **Save block**.
30. Go to the Archived news view `Drupal root/archived-news/` and you will see a standard News article listing, but with a block in the first sidebar which allows you to filter according to year and month.

How it works...

First we create the new view for displaying archived news. To do this we enter a title and a path for the view. We then set what content type we want to output, and the sorting order. We then configure the settings for the page display. We choose to output an unformatted list of Teasers. This means that the list of archived news items will not have any specific formatting, and will be outputted according to the settings in the Teaser view mode, which can be set up in the News article's content type settings.

We then choose to also create a new block with our page; we set this to output an unformatted list of titles. We choose to output 100 items; this is because we want to provide a long list of month and year links for our archive. You may want to set this value higher.

The next stage is where we setup the contextual filtering for the page display. We add the **Content: Created year + month filter**. This means that the page view can then be filtered by adding year and month arguments after the path; for example, to display all articles from December 2011 we filter as follows:

`/archived-news/201112`

The argument follows the pattern `YYYYMM`.

Now that we have a view that can be filtered by month and date, we need to configure the block so that it outputs month and year links for every month that has News article content.

To do this we go to the Block display and open the contextual filter: **Content: Created year + month**. One of the options in the contextual filter is to provide a summary. The summary option means that the view will display a list of all of the created dates associated with the News articles, without repeating the values. Crucially, we then enter **archived-news** in the **Base path** field. This causes the month + year links to begin with `archived-news`, and thus they will link to our Archived news page view.

Finally, we set up the Archived news block so that it displays in the **Sidebar first** region of the theme. We also set it to display only on archived news pages, and any child pages of this URL by adding the asterisk after the path.

Building complex views using relationships

When creating a new View, there are many fields available that can be added to the display. However, where the content type being used in the view has an intrinsic relationship to another content type, a relationship can be added to the View, which allows the View to make use of fields from the related content type. In this recipe we will create a View with a relationship to the User content type, and we will utilize this relationship so that we can output a list of news articles, with the author's profile picture in the display, as illustrated in the following screenshot:

News and author picture

 Sample news article

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis lectus metus, at posuere neque. Sed pharetra nibh eget orci convallis at posuere leo convallis. Sed blandit augue vitae augue scelerisque bibendum. Vivamus sit amet libero...

 Sample news content 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis lectus metus, at posuere neque. Sed pharetra nibh eget orci convallis at posuere leo convallis. Sed blandit augue vitae augue scelerisque bibendum. Vivamus sit amet libero...

 News item 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis lectus metus, at posuere neque. Sed pharetra nibh eget orci convallis at posuere leo convallis. Sed blandit augue vitae augue scelerisque bibendum. Vivamus sit amet libero...

Getting ready

To complete this recipe you will need to have first completed the first recipe in this chapter, *Creating a news listing view*. We will be making use of the News article content type. We will also be making use of the user profile picture field, and upload a profile picture for at least one of the user accounts that has authored a News article node.

To upload a profile picture to a user account, select **People** from the admin menu, then select **Edit** in the row of the user you want to edit. In the **Picture** section, use the **Upload picture** field to choose a picture, then click on **Save**.

To make the display of the profile picture more appropriate to this view, we will be applying an image format called profile-mini. If you would like to apply this image format, create a new image format beforehand, called **user-mini**, with the scale and crop filter that outputs an image at 25x25 pixels. For more information on creating image formats, see *Applying an image format* recipe in *Chapter 4, Custom Content Types*.

How to do it...

We will begin by creating a new View. We will then add a relationship that will allow us to utilize the User data of the News article's author. With the newly available User fields we will output the User's profile picture to appear with the News article listing.

1. Select **Structure** from the admin menu, then select **Views**.
2. Select **+Add new view**.
3. In the **View name** field, enter **News and author picture**.
4. Configure the firstfieldset to read **Show Content of type News article sorted by Newest first**.
5. Ensure that **Create a page** is checked and configure the **Display format** field to read as **Unformatted list of fields with links (allow users to add comments, etc) without comments**.

View name *
News and author picture Machine name: news_and_author_picture [Edit]

Description

Show **Content** of type **News article** tagged with
sorted by **Newest first**

Create a page

Page title

Path

Display format
Unformatted list of **fields** **with links (allow users to add comments, etc.)** **without comments**

Items to display

Use a pager

Create a menu link

Include an RSS feed

6. Click on **Continue & edit**.
7. In the **View edit** page, expand the **Advanced** fieldset.
8. Select the **add** button in the **Relationships** field.

9. In the **Search** field, enter the term **user**.
10. Check the item **Content: Author**.
11. Select **Add and configure relationships**.
12. In the **Configure relationship** popup, leave the values in their default state and select **Apply (all displays)**.
13. In the **Fields** section select **add**.
14. In the **Search** field enter **body**, then check the **Content: body option**.
15. Select **Add and configure field**.
16. In the **Configure field** popup uncheck **Create a label field**.
17. Expand the **Rewrite results** fieldset and check the option **Trim this field to a maximum length**.
18. In the **Maximum length** field enter **250**.
19. Select **Apply (all displays)**.
20. In the **Fields** section select **add**.
21. In the **Search field** enter the term **user picture**.
22. Check the option **User: Picture**, then select **Add and configure fields**.
23. In the **Configure field** popup de-select **Create a label**.
24. In the **Image style** field select **user-mini**.
25. Select **Apply (all displays)**.
26. Select the drop-down next to the **Fields** fieldset, then select rearrange.
27. Arrange the fields so that they are in the following order:
 - (author) User: Picture
 - Content: Title
 - Content: Body
28. Select **Apply (all displays)**.
29. In the **Format** fieldset, select the **Settings** link for the **Show** field.
30. Check the options **(author) User: Picture** and **Content: Title**.

Inline fields	
<input checked="" type="checkbox"/>	(author) User: Picture
<input checked="" type="checkbox"/>	Content: Title
<input type="checkbox"/>	Content: Body

31. Select **Apply (all displays)**.

The screenshot shows the 'Page details' configuration page for a view named 'Page'. The interface is divided into several sections:

- TITLE**: Title: News and author picture.
- FORMAT**: Format: Unformatted list | Settings. Show: Fields | Settings.
- FIELDS**: (author) User: Picture, Content: Title, Content: Body.
- FILTER CRITERIA**: Content: Published (Yes), Content: Type (= News article).
- SORT CRITERIA**: Content: Post date (desc).
- PAGE SETTINGS**: Path: /news-and-author-picture, Menu: No menu, Access: Permission | View published content.
- HEADER**, **FOOTER**, **PAGER** sections.
- Advanced** section: Contextual filters, Relationships, No results behavior, Exposed form, and Other settings.
- view page** button.

32. Select **Save** for the View.

33. Go to `Drupal root/news-and-author-picture` and you will now see a list of News articles with the author's profile picture outputted directly underneath.

How it works...

We begin by creating the new view. We decide to display content of the News article type, and we then decide to display the content as a collection of fields.

In the view configuration we add a relationship. The relationship we add is on the Content author. This means that all of the fields of a particular News article node will be associated with all of the fields of the node's author. Once we have added this relationship, we are able to add all of the author's fields to our display.

Following this we add two more fields to the display. First we add the body text, and trim the output so that it doesn't exceed 250 characters. We then add the newly available **User Picture** field. This will output the author's profile picture, if one is available. We configure the picture field to output according to the user-mini Image format created before beforehand.

After adding the User Picture field, we rearrange the order in which the fields are displayed, so that the image is first, then the title, and finally, the body text. After arranging the fields, we set the first two to display inline. This causes the profile picture and article title to display neatly on one line, with the body text underneath.

There's more...

We have seen how we can extend the range of User fields that are available to the view. However, it is possible to increase this range by using the reference field.

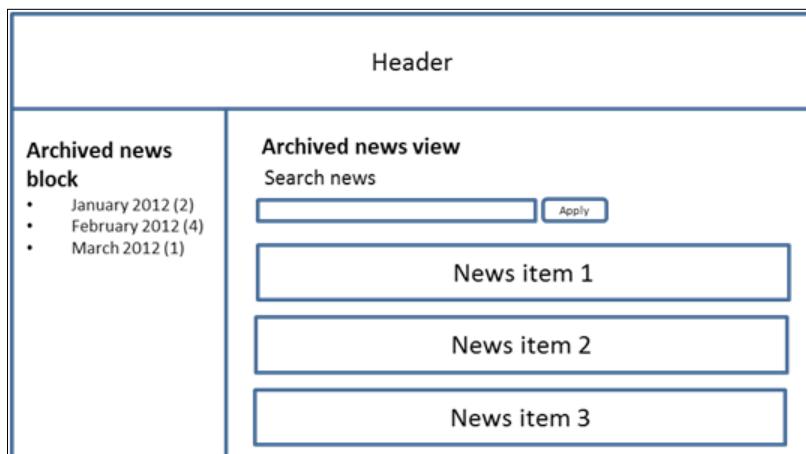
Creating relationships using the reference modules

Using the Node reference module it is possible to include a reference field in a content type where you can reference other content nodes from within a content node. Once this reference has been created, it is then possible to create a relationship in the view on the node reference field. The fields from the referenced node then become available to the view.

For example, you could create two content types, Hotel and Resort. Hotel could hold details such as the hotel name, price per room, and a Node reference field called Resort. The Resort field would be a reference to a Resort node. It would then be possible to create a view that has a relationship on the Resort field of the Hotel content type. The View would then have the capability of outputting a list of Hotel fields describing the hotel, and also some of the Resort fields, made available through the relationship.

Adding a text search filter to a view

We have seen previously in this chapter, how we can filter a view using a block. In this recipe we will modify the news archive view to add a search box to filter the news items by title.



Getting ready

To complete this recipe you will need to have first completed the first recipe in this chapter, *Creating a news listing view*. We will be making use of the News article content type.

This recipe will also be making use of the view created previously in the recipe *Creating an archived content block and view*. Please complete this before proceeding.

How to do it...

We are going to edit the Archived news view and then update the page display so that it exposes a search form for the **Title** field:

1. Select **Structure** from the admin menu, then select **Views**.
2. Select **edit** for the **Archived news** view.
3. In the **Filter criteria** section select **add**.
4. Choose **This page (override)** from the **For** drop-down.
5. Check the option **Content: Title (title)**, then select **Apply (this display)**.
6. In the **Configure filter criterion** popup check the option **Expose this filter to visitors, to allow them to change it**.
7. In the **Label** field change **Title** to **Search news**.
8. Select **Apply (this display)**.
9. Click on **Save**.
10. Now go to Drupal root/archived-news and you will see the search attachment form above the archived news page view.

The screenshot shows the 'Archived news' view page. On the left, there's a sidebar with a 'Search news' input field and an 'Apply' button. Below that is a list of filters: 'November 2011 (1)' and 'December 2011 (3)'. The main content area displays a news article titled 'Sample news article'. It includes a small thumbnail image of a rainbow over a city skyline, the author's name 'dylan', and the submission date 'Fri, 12/23/2011 - 15:06'. A large amount of placeholder text follows the article summary.

Read more

How it works...

We begin by editing the existing Archived news view. We add a filter view to our archived news view. We choose to add the **Content: Title** filter. We check the option to expose the filter. This means that instead of statically entering the text to filter by, in the view, we let the user enter the text themselves.

Using attachments to extend Views' output

In this chapter, we have seen how we can create a View to output lists of nodes of a particular content type. This is perfectly adequate for most uses, but it can become slightly repetitive when you have many listings on a site. In this recipe we will be creating a view that has two displays, one which displays the three most recent news articles, each with a thumbnail image, a title, and some body text. We will then create another display to output a further 10 News articles, but in a less detailed format. Using the technique learnt in this recipe you will be able to create a multitude of Views with mixed formats suitable for pages such as homepages or landing pages, where plain list views are not quite enough.

Getting ready

To complete this recipe you will need to have first completed the first recipe in this chapter, *Creating a news listing view*. We will be making use of the News article content type.

How to do it...

We will begin by creating a new View to which we will add **Image** and **Body** fields. We will configure the Page display to output three of the most recent News articles in a detailed format, and then add an Attachment display to output a further 10 News articles, which will be displayed in a less detailed format.

1. Select **Structure** from the admin menu, then select **Views**.
2. Select **+Add new view**.
3. In the **View name** field enter **News landing page**.
4. Configure the firstfieldset to read **Show Content of type News article sorted by Newest first**.

5. In the **Create a page** fieldset set the **Display format** to **Table**:

The screenshot shows the 'Create a page' configuration form. At the top, there's a 'View name *' field containing 'News landing page' and a 'Machine name' field showing 'news_landing_page' with an '[Edit]' link. Below this is a 'Description' section with a checkbox. Underneath are settings for 'Show' (Content), 'of type' (News article), and 'sorted by' (Newest first). A large 'Create a page' section follows, containing:

- Page title:** News landing page
- Path:** http://drupal7cbc1.streetfish.co.uk/ news-landing-page
- Display format:** Table (selected) of fields
- Items to display:** 3
- Options:** Use a pager, Create a menu link, Include an RSS feed.

6. Click on **Continue & edit**.
7. In the **Fields** section, select **Add**.
8. Set the **For** drop-down to **This page (override)**.
9. In the **Search** field enter the term **image**.
10. Check the option **Content: Image**; ensure it's the one associated with **node:news_article**.
11. Change the **Search** field term to **body**, and check the option **Content: Body**, then select **Apply (this display)**.
12. In the **Configure field** popup, uncheck the option **Create a label**.
13. Set the **Formatter** field to **Trimmed**, then enter **400** in the **Trim length** field.
14. Select **Apply (this display)**.
15. In the **Image** field's **Configure** field popup uncheck the option **Create a label**.
16. In the **Image style** field, select the **Thumbnail** format.
17. In the **Link image to** field, select **Content**.
18. Select **Apply (this display)**.
19. In the **Fields** section, select the drop-down next to the **add** button, then select **Rearrange**.
20. Ensure that **This page (override)** is selected in the **For** field.
21. Drag the **Content: Image** row to the top of the list.

22. Select **Apply (this display)**.
23. In the **Format** section, select the **Settings** link.
24. Select **This page (override)** in the **For** field drop-down.
25. In the **Content: Body** row, change the **Column** drop-down to **Content: Title**.
26. In the **Content: Title** row, enter **
** in the **Separator** column.
27. Select **Apply (this display)**.

The screenshot shows the 'Page' view configuration screen. At the top, there are tabs for 'Page', 'Attachment' (which is circled in red), and '+ Add'. Below the tabs, there's an 'edit view name/description' button. The main area is titled 'Page details' and contains several sections:

- TITLE**: Title: News landing page
- FORMAT**: Format: Table / Settings
- FIELDS**: Content: Image, Content: Title, Content: Body
- FILTER CRITERIA**: Content: Published (Yes), Content: Type (= News article)
- SORT CRITERIA**: Content: Post date (desc)
- PAGE SETTINGS**: Path: /news-landing-page, Menu: No menu, Access: Permission | View published content
- HEADER**
- FOOTER**
- PAGER**: Use pager: Display a specified number of items / 3 items, More link: No
- Advanced** section: Contextual filters, Relationships, No results behavior
- EXPOSED FORM**: Exposed form in block: No, Exposed form style: Basic | Settings
- OTHER**: Machine Name: page, Comment: No comment, Use AJAX: No, Hide attachments in summary: No, Use aggregation: No, Query settings: Settings, Field Language: Current user's language, Caching: None, CSS class: None, Theme: Information

28. Select **+Add** then select **Attachment** from the drop-down:



29. In the **Fields** section, select the field **Content: Image**.
30. In the **Configure field** popup, ensure that **This page (override)** is selected, and then select **Remove**.
31. In the **Pager** section, select the second link in the **Use pager** field to open the **Pager options** popup.
32. Select **This page (override)** in the **For** field.
33. Change the **Items per page** to **10** and set the **Offset** to **3**.
34. Select **Apply (this display)**.
35. In the **Attachments settings** section select the link next to **Attach to**.
36. In the **Attach to** popup check the **Page** option and select **Apply**.
37. In the **Attachments settings** section select the link next to the **Attachment position** field.
38. Select the **After** option and select **Apply**.

The screenshot shows the 'Attachment' view configuration page. At the top, there are tabs for 'Page', 'Attachment' (which is selected), and '+ Add'. To the right is a button for 'edit view name/description' and a dropdown menu. Below the tabs, the title 'Attachment details' is expanded. The configuration is organized into several sections:

- TITLE**: Title: News landing page
- FORMAT**: Format: Table | Settings
- FIELDS**: Content: Title (with an 'add' button)
- FILTER CRITERIA**: Content: Published (Yes) | Content: Type (= News article) (with an 'add' button)
- SORT CRITERIA**: Content: Post date (desc) (with an 'add' button)
- ATTACHMENT SETTINGS**: Attach to: Page | Attachment position: After | Inherit contextual filters: Yes | Inherit exposed filters: No | Access: Permission | View published content
- PAGER**: Items to display: Display a specified number of items | 10 items, skip 3 | More link: No | Inherit pager: No | Render pager: No
- HEADER** and **FOOTER** sections with 'add' buttons
- Advanced** section:
 - CONTEXTUAL FILTERS**: add
 - RELATIONSHIPS**: add
 - NO RESULTS BEHAVIOR**: add
 - EXPOSED FORM**: Exposed form style: Basic | Settings
 - OTHER**: Machine Name: attachment_1 | Comment: No comment | Use AJAX: No | Use aggregation: No | Query settings: Settings | Field Language: Current user's language | Caching: None | Link display: Page | CSS class: None | Theme: Information

39. Click on **Save**.
40. Go to Drupal root/news-landing-page and you will see a news listing with three headline items, and a list of 10 of the next articles in a less detailed format. To fully test this recipe you will need to have added at least 13 News article nodes.

How it works...

We begin by creating a new view to display the Landing page news. We choose to display content of the News article type, and we decide to set the **Display format** to **Table**. Setting the **Display format** to **Table** will mean that we must output a collection of fields that we will need to set up on the following screen. We set the number of items to display as 3. This is because the first display we are creating will only output the top three articles, and they will be displayed with more detail. Later on we will add an attachment to display a further 10 articles, but with less detail.

We now begin to configure the fields. We already have the default **Title** field, so we add the **Content: Image** field associated with the News article content type, and the **Content: Body** field. We first configure the **Body** field so that the text is trimmed to a maximum of 400 characters. We then configure the **Image** field so that the **Image format** is set to **Thumbnail**. This will ensure that the image is outputted at the predetermined Thumbnail size. We also set the image to link to the content page of the News article.

After adding the new fields, they need to be rearranged to output in the correct order. We select the **Rearrange** option in the **Fields** drop-down and move the **Image** field above the others so that it is outputted first.

Following this we update the **Table format** settings. We change the **Column settings** for the row **Content: Body** and set it to **Content: Title**. This causes the output of the **Content: Body** field to appear in the same column as the **Content: Title** field. Without doing this, the body field would appear in a column of its own. We also set the **Separator** for the Title field to a **
** to ensure that the Body text appears underneath the Title.

We now add a new **Attachment** display, which will be used for outputting the next 10 News articles, but in a less-detailed format. We first remove the **Image field** from the display, and then update the **Pager** settings to display 10 items. We set the **Offset** field to 3. This causes the attachment listing to skip past the first three News articles, and start at the fourth. We do this because the **Page** display is already displaying the first three items.

Finally, we set the **Attach to** settings so that the Attachment is attached to the Page display, and in the **Attachment position** settings, we set the Attachment to display below the content of the **Page** display.

6

Creating Flexible Pages Using Panels

In this chapter we will cover:

- ▶ Adding custom text to a page
- ▶ Adding a block to a page
- ▶ Adding a dynamic view to a page
- ▶ Configuring the visibility of the page
- ▶ Creating a custom page layout using the Layout builder

Introduction

In this chapter we will introduce the **Panels** module. Panels is used to create pages consisting of one or more items of content of various types, including views, blocks, nodes, and menus.

There are a number of factors to consider when deciding whether to use a Panel or not. It is perfectly acceptable to simply use the blocks system to display content blocks at various regions on a regular node display. However, Panels adds a great deal of flexibility, by allowing you to add much more than blocks, but at the small price of increasing the load time, and complexity of the HTML output.

Using Panels, you can add a wide variety of content such as views, blocks, content nodes, and forums to any region of a panel, and the site editor can easily move them around. There are a number of different layouts that can be used, and the site editor can even create their own, from within the CMS.

The homepage of a site is the perfect candidate for using a Panel. There are usually lots of distinct items, such as news blocks, maybe a carousel view, and some introductory text. However, you will probably not want to use a panel for regular run-of-the-mill content pages as it really is overkill, and it will increase the complexity of the site.

This chapter begins with a simple recipe demonstrating how to create a page with one item of custom text. Following this, the chapter progresses on to recipes that describe how to create pages that contain blocks and also dynamic views, where an argument is passed into the view from the page.

Towards the end of the chapter, we will see how a page can be configured so that its visibility is restricted according to the role of the logged in user. Finally, we will see how to create a custom layout using the Layout builder.

Getting ready

To complete the recipes in this module, you will need to install the latest recommended releases of the following modules:

- ▶ <http://drupal.org/project/panels>
- ▶ <http://drupal.org/project/ctools>
- ▶ <http://drupal.org/project/views>

You will need to enable, at least, the following features of the above modules:

- ▶ Chaos tools
- ▶ Page manager
- ▶ Views content panes
- ▶ Panels
- ▶ Views
- ▶ Views UI

Adding custom text to a page

Pages provide a flexible means of displaying various types of content on a page. In this recipe we will be creating a new page with a two column layout and then adding a new item of custom content to it.

Getting ready

For this recipe, please ensure you have installed the **Panels** and **Views** modules described in the introduction to this chapter.

How to do it...

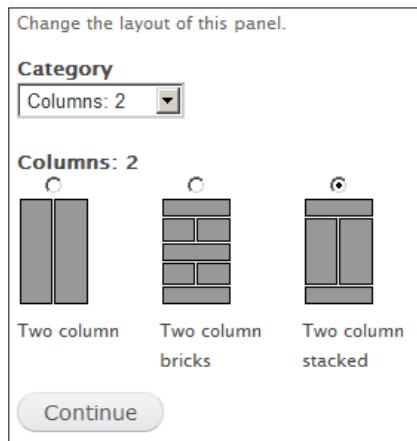
To complete this recipe we will first create a new **Page** and select its layout. Then we add a **New custom content** item to the **Top region** of the page. Finally, we will save and test the page.

1. Select **Structure** from the admin menu, then select **Pages**.
2. Select **+Add custom page**.
3. In the **Administrative title** field enter **Custom text page**.
4. In the **Path** field enter `pages/custom-text`.
5. Leave all other options in their default state and click on **Continue**.

Administrative title <input type="text" value="Custom text page"/> Machine name: <code>custom_text_page</code> [Edit] The name of this page. This will appear in the administrative interface to easily identify it.
Administrative description <input type="text"/> A description of what this page is, does or is for, for administrative use.
Path <input type="text" value="http://drupal7cbc1.streetfish.co.uk/ pages/custom-text"/> The URL path to get to this page. You may create named placeholders for variable parts of the path by using <code>%name</code> for required elements and <code>!name</code> for optional elements. For example: "node/%node/foo", "forum/%forum" or "dashboard/!input". These named placeholders can be turned into contexts on the arguments form.

6. On the **Choose layout** page, select **Columns: 2** from the **Category** field.

7. Select the option **Two column stacked**:



8. Click on **Continue**.
9. On the **Panel settings** page leave the options in their default states and click on **Continue**.
10. On the **Panel content** page, select the gear for the **Top** region of the page, then choose the option **Add content**.
11. In the **Add content** popup, select the option **New custom content**.
12. In the popup, set the **Administrative title** to **Custom text content**.
13. Leave the **Title** field blank.
14. In the **Body** field enter **This is some custom text**:

The dialog box has sections for 'Administrative title' (set to 'Custom text content'), 'Title' (empty), 'Body' (containing 'This is some custom text'), and 'Text format' (set to 'Plain text'). A note below the text format says: 'No HTML tags allowed. Web page addresses and e-mail addresses turn into links automatically. Lines and paragraphs break automatically.' At the bottom are 'Finish' and 'Cancel' buttons.

15. Click on **Finish**.
16. You will now see the **Panel content** page with the new Custom text item added to the **Top** region:



17. Click on **Finish**, then on **Save** to finish adding the page to the database.
18. Go to `Drupal root/pages/custom-text` to view the new page.

How it works...

We begin by creating a new custom page. To do this we enter an **Administrative title** for identifying the page in the pages list, then we enter a **Path** that can be used for accessing the page. We then decide the layout for the new page. In this case we select the **Two columns stacked** layout; there are a number of different layouts that can be used for creating a wide variety of configurations.

After completing the layout settings and general panel settings we begin to add content to the page. Each layout region of the page has a gear button that provides the option to add many types of content to the page, including preexisting views, blocks, and nodes. In this case we add a **New custom content** item to which we add an administrative title and some sample body content. We then save and test the new page.

There's more...

This recipe describes the simplest method of adding content to a page because content can be added directly to the page without first creating a new node. However, adding an existing node to the page is not too complicated.

Adding an existing node to a page

The method in this recipe shows how you can create a new item of content on-the-fly without having to create a node beforehand. This is suitable for simple content which probably will not need to be reused. However, if you are dealing with an item of content that is likely to be periodically updated, or reused elsewhere, it's preferable to create a content node.

To add an existing node, first create the new node as described in the *Creating a basic page and adding it to the main menu* recipe in *Chapter 2, Creating and Publishing Content*. After the node has been created, it will then be possible to add the node to the page using the **Add content** popup, and then selecting **Existing node**, instead of **New custom content**. It is then possible to search for and select an existing node using the autocomplete field.

Adding a block to a page

In the preceding recipe we saw how you can add custom content text to a page. However, a page can really come to life when it contains varied items of content. In this recipe, we will learn how to create a simple block using **Views** and then add that block to a page.

Getting ready

For this recipe, please ensure you have installed the **Panels** and **Views** modules described in the introduction to this chapter.

How to do it...

We begin this recipe by creating a new **View** that outputs one block display of the five latest items of content. We will then create a **new Page** to which we will add the **Latest content** block. Finally, we will save our new page and review the new page.

1. Select **Structure** from the admin menu, then select **Views**.
2. Select **+Add new view**.
3. In the **View name** field enter **Latest content**.
4. Uncheck the **Create a page** field and check the **Create a block** field.

5. Leave the **Create a block** field options in their default states click on **Save & exit**:

The screenshot shows the 'Create a block' configuration page. At the top, there is a 'View name *' field containing 'Latest content' and a note 'Machine name: latest_content [Edit]'. Below it is a 'Description' checkbox. Under the heading 'Show', there are dropdown menus for 'Content' (selected), 'of type' (All), and 'tagged with' (radio button selected). A 'sorted by' dropdown menu is set to 'Newest first'. There are two checkboxes: one for 'Create a page' (unchecked) and one for 'Create a block' (checked). Under 'Create a block', there is a 'Block title' field with 'Latest content', a 'Display format' section with 'Unformatted list' and 'titles (linked)', an 'Items per page' input field with '5', and a 'Use a pager' checkbox (unchecked). At the bottom are three buttons: 'Save & exit' (highlighted in grey), 'Continue & edit', and 'Cancel'.

6. Select **Structure** from the admin menu, then select **Pages**.
7. Select **+Add custom page**.
8. In the **Administrative title** field enter **Page with block**.
9. In the **Path** field enter **pages/block-page**.
10. Leave all other options in their default state and click on **Continue**.
11. On the **Choose layout** page, select **Miscellaneous** from the **Category** field.
12. Select the option **Three column 33/34/33 stacked**.
13. Click on **Continue**.
14. On the **Panel settings** page leave the options in their default states and click on **Continue**.
15. Select the gear in the **Left side** region of the panel, then select **Add content** from the drop-down.
16. Select **Views** from the tab, then select **Latest content**.

17. In the **Select display** popup select **Block** from the **Display** drop-down, then click on **Continue**.
18. In the **Configuration** popup for the block, leave the default options and click on **Finish**.

The screenshot shows the 'Panel configuration' page. At the top, there's a 'Title type' dropdown set to 'Manually set'. Below it is a 'Title' input field which is empty. The main area is divided into 'Top' and 'Bottom' sections. The 'Top' section contains three columns: 'Left side', 'Middle column', and 'Right side'. The 'Left side' column is currently selected, indicated by a dashed border. Inside this column, there's a 'View: Latest content: Block' configuration with a gear icon and a 'View information' link. The 'Middle column' and 'Right side' columns are also visible but not selected. Below the 'Top' section is a 'Bottom' section containing two buttons: 'Back' and 'Finish'.

19. Back in the **Panel configuration** page, click on **Finish**, then on **Save**.
20. Go to `root/pages/block-page` to see your new page with the **Latest content block**.

How it works...

We begin by creating a new view. We configure the view to display only a block display, and we use the default options in the block setup to display an **Unformatted list of Titles**, 5 per page. This is all we need to create our simple block for the new page.

Next we create a new **Custom page**, setting the **Administrative title** to **Page with block** so that we can identify the page in the **Page manager**. We then add a path where the page can be accessed. Following this, we set up the layout for the new page. In this recipe, we set the layout to **Three column 33/34/33 stacked**, which means that the page will have three columns with widths of 33%, 34%, and 33%, with a row above and below.

After configuring the layout and settings we add content to the page. To do this we find the region that we want to add content to, in this case the **Left** side, then select the gear icon to open the drop-down, then we select the **Add content** option. In the popup we select the **Views** tab and select **Latest content** view that we created previously. We do not need to configure any options for the view on this occasion so we **Save** our changes and view the result.

Adding a dynamic view to a page

One of the important features that makes the Panels module so flexible is that it allows us to add views to a page's regions. In this recipe, we will not only be adding a preexisting view to a page, but we will be supplying an argument to the view, via the panel, to filter it.

In this recipe, we will be making use of the **Glossary** view, which is one of the sample views shipped with the **Views** module. The **Glossary** view accepts an alphanumeric argument to filter the view to display glossary terms for the supplied letter, which is why it's termed a dynamic view.

This recipe illustrates the way in which a dynamic view can have its argument supplied through the Panel, meaning that a view can be recycled on different pages, just by having different arguments supplied to it.

The outcome of this recipe is a page displaying a glossary listing filtered by the letter "a", as follows:

A (1) B (1) C (2) D (1) G (2) M (2) N (2) O (2) P (1) S (4) T (8)		
Title ▲	Author	Last update
About us	dylan	Wed, 04/18/2012 - 22:11

Getting ready

For this recipe, please ensure you have installed the **Panels** and **Views** modules described in the introduction to this chapter.

How to do it...

We will begin by enabling the **Glossary** view, which is a sample view provided with the **Views** module. We will then create a new page to which we will add the glossary view. When we add the glossary view onto the panel, we will set an argument which will be passed to the view. This argument will be used by the view to filter the output of the glossary.

1. Select **Structure** from the admin menu, then select **Views**.
2. Select **enable** for the **Glossary** view.
3. Select **Structure** from the admin menu, then select **Pages**.
4. Select **+Add custom** page.
5. In the new **Page** popup, enter **Page with dynamic Glossary view** in the **Administrative title** field.
6. In the **Path** field enter `pages/filtered-glossary`.
7. Leave the other options in their default state and click on **Continue**.

8. In the **Choose layout** page, select **Columns: 1** from the **Category** drop-down, then select the **Single column** option.
9. Click on **Continue**.
10. On the **Panel settings** page leave the options in their default state and click on **Continue**.
11. Select the gear icon for the **Middle column** region, then select the **Add content** option from the drop-down.
12. Select the **Views** tab in the **Add content** popup.
13. Select the **Glossary** view from the list of views:

Add content to Middle column

Activity	<input checked="" type="checkbox"/> Archived news	<input checked="" type="checkbox"/> News
Custom blocks	<input checked="" type="checkbox"/> Basic page RSS feed	<input checked="" type="checkbox"/> News and author picture
Menus	<input checked="" type="checkbox"/> feeds log	<input checked="" type="checkbox"/> News landing page
Miscellaneous	<input checked="" type="checkbox"/> Glossary	<input checked="" type="checkbox"/> Recent content listings
Page elements	<input checked="" type="checkbox"/> Latest content	
Views		
Widgets		
<input checked="" type="checkbox"/> New custom content		
<input checked="" type="checkbox"/> Existing node		

14. In the Select display popup select **Page** from the **Display** drop-down, then click on **Continue**.
15. In the **Configure view** popup check the field **Send arguments**.
16. In the **Arguments** field, enter the letter **a**:

Send arguments

Select this to send all arguments from the panel directly to the view. If checked, the panel arguments will come after any context arguments above and precede any additional arguments passed in through the Arguments field below. Note that arguments do not include the base URL; only values after the URL or set as placeholders are considered arguments.

Arguments

a

Additional arguments to send to the view as if they were part of the URL in the form of arg1/arg2/arg3. You may use %0, %1, ..., %N to grab arguments from the URL. Or use @0, @1, @2, ..., @N to use arguments passed into the panel. Note: use these values only as a last resort. In future versions of Panels these may go away.

17. Click on **Finish**.

The screenshot shows the 'Basic settings' path followed by 'Choose layout', 'Panel settings', and finally 'Panel content'. In the 'Title type' section, 'Manually set' is selected. The 'Title' field is empty. Below it, a note says: 'The title of this panel. If left blank, a default title may be used. Set to No Title if you want the title to actually be blank.' The 'Middle column' region contains a 'View: Glossary: Page *' configuration. This configuration includes a 'View information' link and two buttons at the bottom: 'Back' and 'Finish'.

18. On the **Panel Content** page click on **Finish**, then on **Save**.

19. Go to the URL `Drupal root/pages/filtered-glossary/` to review the finished page; notice that the page has been filtered to display only nodes beginning with **a**.

How it works...

We first enable the **Glossary** view that comes with the **Views** module. This is a sample view that will display all content nodes known to the system, grouped by starting letter. It contains a dynamic filter that allows it to be dynamically set to display nodes beginning with a particular argument when it is loaded.

We then create a page to display the filtered glossary on. When creating a new page we add an **Administrative title** so we can identify the page when looking through the **Pages list**, and we add a **Path** where we can access the page from.

We now decide on the layout for the page. To do this we select the option **Columns: 1** from the **Category** drop-down. We then choose the **Single column** option, which has only one region, **Middle column**.

The next step is to add content to the page. To do this we select **Add content** from the gear drop-down in the **Middle column** region. In the **New content** popup, we select the **View** tab to display the list of available views, then select the **Glossary** view.

The **Select display** popup is where we choose which of the view's displays we want to use. In this case, the **Page** display.

On the **Configure** view popup, we specify which argument is to be sent to the view. We check the **Send arguments** option to ensure that arguments are sent, and then we specify the argument we want to send in the **Arguments** field. The argument "a" will be sent to the **Glossary** view as if it had been appended to the URL.

After adding the content we review the page.

Configuring the visibility of the page

There are many situations that will require a page's visibility to be restricted in some way. For example, a private members area to display restricted content. In this recipe, we will see how a page can be setup with restrictions that make it accessible by a specified user role.

Getting ready

For this recipe, please ensure you have installed the **Panels** and **Views** modules described in the introduction to this chapter.

How to do it...

We will begin by creating a new page to which we will add some sample content. We will then configure the access permissions so that only authenticated users can access the page:

1. Select **Structure** from the admin menu, then select **Pages**.
2. Select **+Add custom page**.
3. In the **Administrative title** field enter **Member's only page**.
4. In the **Path** field enter `pages/members-only-page`.
5. Click on **Continue**.
6. In the **Choose layout** page, select **Columns: 2** from the **Category** drop-down.
7. Select the **Two column bricks** option, then click on **Continue**.
8. Leave the **Panel settings** page in its default state and click on **Continue**.
9. On the **Panel content** page, select the gear in the **Top region**, then select **Add content** from the drop-down.
10. Select the option **New custom content**.
11. In the **New custom content** popup enter **Members only content** in the **Administrative title** field.

12. In the **Body** field, enter **This is a members only page containing secret information. Only authenticated users can see this page.**
13. Click on **Finish** and then in the **Panel content** popup select **Finish** again.
14. Select the **Access** button from the left-hand side menu.
15. From the drop-down select **User: role** and then click on **Add:**

The screenshot shows the 'Access' settings page for a specific page in a Drupal 8 administration interface. The page title is 'Member's only page'. The left sidebar has a 'Settings' section with 'Access' highlighted. The main content area is titled 'Settings > Access' and contains a table with one row: 'No criteria selected, this test will pass.' Below the table is a dropdown menu set to 'User: role' with an 'Add' button next to it. At the bottom, there are two radio buttons: 'All criteria must pass.' (selected) and 'Only one criteria must pass.' followed by 'Update' and 'Update and save' buttons. A message at the bottom says: 'You have unsaved changes to this page. You must select Save to write them to the database, or Cancel to discard these changes. Please note that if you have changed any form, you must submit that form before saving.' There are 'Save' and 'Cancel' buttons at the bottom.

16. In the **Add criteria** popup check the role **authenticated user**, then click on **Save**:

The screenshot shows a configuration dialog for a user role. At the top, it says "User" and "Logged in user". Below that is a "Role" section containing three checkboxes: "anonymous user" (unchecked), "authenticated user" (checked), and "administrator" (unchecked). A note below the checkboxes states "Only the checked roles will be granted access." At the bottom is a "Save" button.

17. On the **Access settings** page, ensure that **All criteria must pass** is selected, then click on **Save**.
18. To test, visit the URL `Drupal root/pages/members-only-page` while logged in and you will see the members area page.
19. Log out and visit the page again; you will encounter the **Access denied message**.

How it works...

This recipe begins with creating a new page. The page is given an **Administrative title** and a **Path** where it can be accessed. Next, we select the layout for the page. In this case we are selecting a two column layout called **Two-column bricks**. Once the layout has been decided we add the new custom content.

Once the content is added we open the **Access settings** page where we add the **User: role** criterion to the access criteria list. We then specify the **authenticated user** role. This means that the user viewing this page must, at least, have the authenticated user role applied to them. Finally, we test our new page while logged as an **authenticated user**, and while logged out, to demonstrate that the page is not accessible without the **authenticated user** role.

There's more...

In this recipe, we have seen how to limit the visibility using just one criterion, the user role. However, there are other criteria that can be used in conjunction with each other.

Other access criteria

In addition to using the User's role as an access criterion, it is possible to use a combination of other criteria, including the following:

- ▶ Access dependent on current theme
- ▶ Access depending on current URL
- ▶ Access depending on whether a supplied snippet of PHP code returns true

Creating a custom page layout using the Layout builder

The **Panels** module comes equipped with various prebuilt layouts. While the existing layouts are suitable for many different uses, it is possible to use the Layout builder to build a bespoke solution. In this recipe, we will be building a page with a full width top row region with four column regions below.

Getting ready

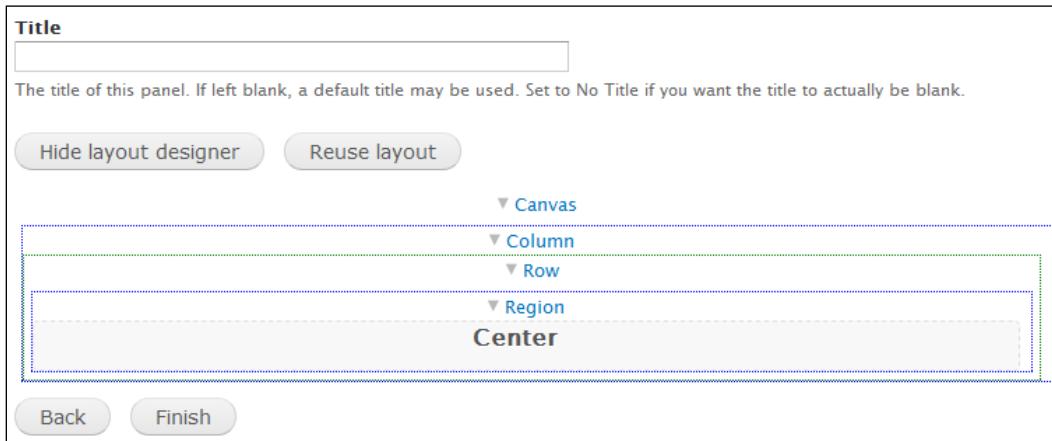
For this recipe, please ensure you have installed the **Panels** and **Views** modules described in the introduction to this chapter.

How to do it...

We will first be creating a new sample **Page** called **Custom layout** page, we will then set the **Layout** to **Flexible** and finally we will create a custom layout using the Layout builder:

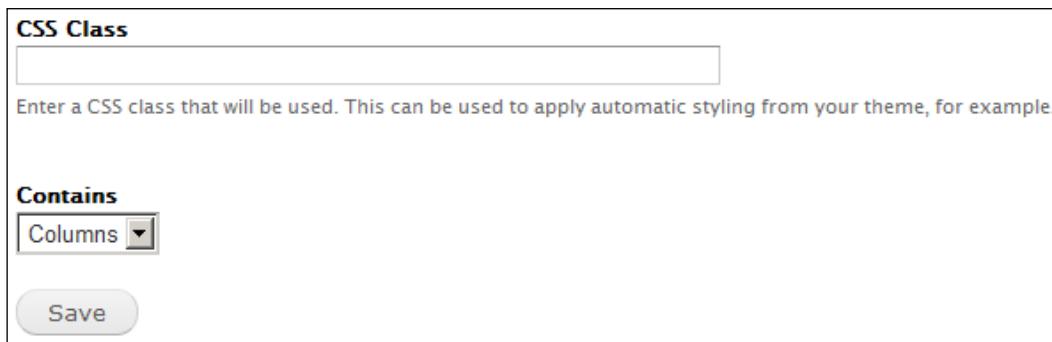
1. Select **Structure** from the admin menu, then select **Pages**.
2. Select **+Add custom page**.
3. In the **Administrative title** field enter **Custom layout page**.
4. In the **Path** field enter `pages/custom-layout-page`.
5. Click on **Continue**.
6. On the **Choose layout** page, select **Builders** from the **Category** drop-down.
7. Select the **Flexible** option, then click on **Continue**.
8. On the **Panel settings** page, leave the settings in their default state and click on **Continue**.

9. Select **Show layout designer** to go to **Switch to the editing mode**.



10. Select the **Column** drop-down, then select **Add row to bottom**.

11. In the **Add row** popup, select **Columns** from the **Contains** field, then click on **Save**:



12. Select the lower **Row** drop-down, then choose the **Add column** option.

13. In the **Add column** popup, ensure **Fluid** is selected in the **Width** field, then click on **Save**:

CSS Class
Enter a CSS class that will be used. This can be used to apply automatic styling from your theme, for example.

Width
Fluid

Save

14. Select the lower **Row** drop-down again, then choose the **Add column to right** option.
15. In the **Add column** popup, ensure **Fluid** is selected in the **Width** field, then click on **Save**.
16. Repeat steps 14-15 so you end up with four columns
17. Drag each of the dragbars for the four columns so that each column receives **25%** of the width (or as close to 25% as it will allow).
18. Click on **Finish** and select the **Show layout designer** button.
19. In the lower **Row**, select the first **Column** drop-down, then select **Add row**.
20. In the **Add row** popup, select **Regions** from the **Contains** field, then click on **Save**:

CSS Class
Enter a CSS class that will be used. This can be used to apply automatic styling from your theme, for example.

Contains
Regions

Save

21. In the lower row, select the **Row** drop-down in the first **Column** drop-down, then choose the **Add region** option.
22. In the **Add region** popup, enter **Column 1** in the **Region title** field.

23. Ensure that the **Width** field is set to **Fluid**, then click on **Save**:

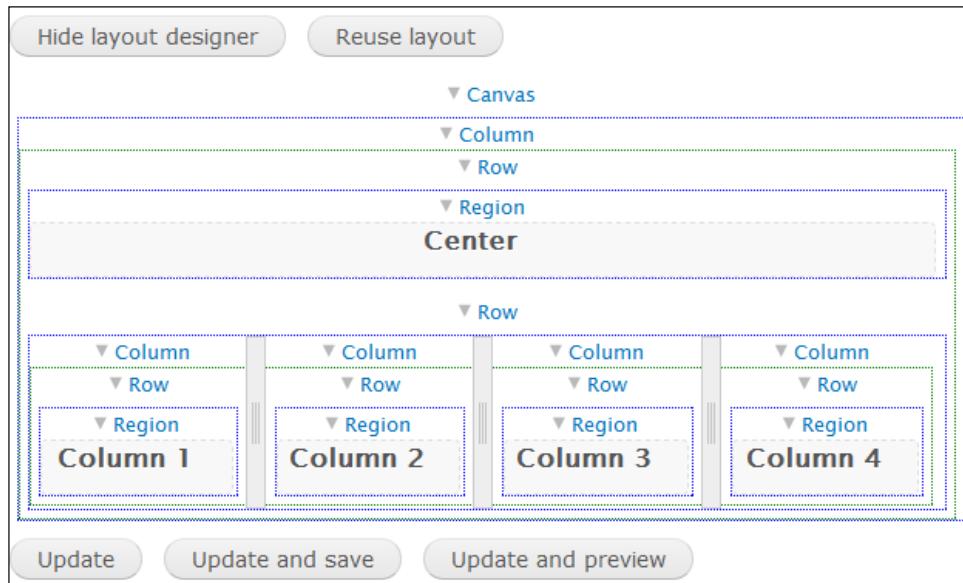
Region title *
Column 1

CSS Class

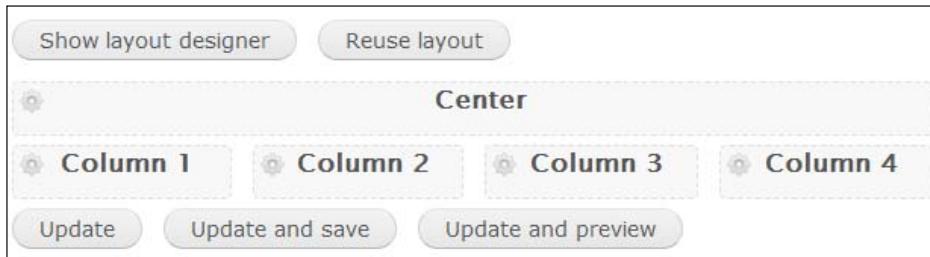
Enter a CSS class that will be used. This can be used to apply automatic styling from your theme, for example.

Width
Fluid

24. In the lower **Row**, select the second **Column** drop-down, then select **Add row**.
25. In the **Add row** popup, select **Regions** from the **Contains** field, then click on **Save**.
26. Repeat steps 22-25, but add the new row to the **second** column, and set the new Region title to **Column 2**.
27. Repeat steps 22-25, but add the new row to the **third** column, and set the new Region title to **Column 3**.
28. Repeat steps 22-25, but add the new row to the **fourth** column, and set the new Region title to **Column 4**.
29. Click on **Update and save**.



30. Select the **Hide layout designer** button to display the regions which are now ready to have content added to them.



31. Click on **Save** to finish.

How it works...

We begin by creating a new **Page**. We then select the **Flexible layout**; this enables the use of the Layout builder. The Layout builder consists of the Canvas to which, any number of columns can be added. Rows can then be added to any of the columns. Finally, after creating the rows and columns, the regions must be added and named.

To begin creating our flexible layout, we select **Show layout designer**, which displays the layout editing mode. We then add a new row to the bottom of the layout. To this row we then add four columns. After adding the columns, we set their widths to 25% each, or as close to 25% as possible, using the drag bar.

After adding the four columns we add a row to each of these new columns. We have to add a row to them because regions can only be added to rows.

There's more...

We have seen how we can use the Layout designer to create many different kinds of page layout. After working hard on a new layout it would be a shame to lose it. Luckily, there's a way to save it for reuse.

Saving a custom layout for reuse

There is a button on the **Content page** called **Reuse layout**. Selecting this will open a popup where it is possible to name and categorize the layout, and then to save it. After saving the layout, it can be selected from the list of available layouts, just as the other layouts are selected.

7

Working with Media

In this chapter we will cover:

- ▶ Creating a simple slideshow carousel
- ▶ Creating a document content type
- ▶ Creating a simple document library
- ▶ Linking documents to a content type
- ▶ Adding video to a content type

Introduction

In this chapter, we will delve into the various methods of adding media to your content. In *Chapter 2, Creating and Publishing Content*, we saw how we can add media to content using the WYSIWYG editor. The WYSIWYG editor can be used to add images, documents, and videos into your content. However, this method leaves a great deal of control to the user adding the content. This is sometimes undesirable, as it can lead to inconsistency.

For example, if you wanted to run a video blog, you could allow users to enter the video embed code directly into the body text using the WYSIWYG editor, but this could lead to the video being displayed before, after, or between the text. In this chapter, we will see how to provide specific fields for attaching media to content, so that we can tightly control how the media is displayed.

This chapter begins by exploring how to build a simple slideshow carousel that is still powerful and flexible enough so that it can be very easily adapted to fit any size requirements.

Following the slideshows, we begin to create a document library. First, there is a recipe that deals exclusively with creating a document content type that manages the file upload. We then build upon this and create a view which displays the list of documents. Finally, there is a further recipe that shows a method for adding links to files in the document library, from another content type.

We finish the chapter with a recipe that deals with embedding video onto a page. Unlike embedding video into a regular Body field, this method provides a way to add video embed code to its own field, in a structured and controlled fashion.

Creating a simple slideshow carousel

In this chapter, we will be using a set of contributed modules to create a simple slideshow content type.

Simple carousel



[Prev](#) [Next](#)

Getting ready

For this recipe you will need to install and enable the most recent recommended releases of the following modules:

- ▶ http://drupal.org/project/field_slideshow
- ▶ <http://drupal.org/project/jcarousel>
- ▶ <http://drupal.org/project/libraries>

You will need FTP access to your site for installing the jQuery cycle plugin.

How to do it...

First we will install the jQuery cycle plugin, then we will create a new content type for the slideshow, and finally we will configure the display of the slideshow and add some test slides:

1. Open your FTP client and create a new directory called **libraries** in `Drupal root/sites/all/` if it does not already exist.
2. In the **libraries** directory create a new folder called `jquery.cycle`.
3. Download the jQuery cycle plugin from <http://jquery.malsup.com/cycle/download.html>, be sure to get the full cycle plugin, and not the lite version.
4. Back on the website, select **Configuration** from the admin menu, then select **Image styles**.
5. Select **+Add style**.
6. In the **Image styles** popup enter **simple-slideshow** in the **Style name** field.
7. On the **Edit image style** popup, select the **Select new effect** drop-down, then choose the option: **Scale and crop**.
8. Click on **Add**.
9. In the **Width** field enter **700**.
10. In the **Height** field enter **300**.
11. Click on **Add effect**.
12. Select **Structure** from the admin menu, then select **Content types**.
13. Select **+Add content type**.
14. Enter **Simple slideshow** in the **Name** field.
15. Select the **Display settings** tab, then uncheck the **Display author and date information**.
16. Select the **Comment settings** tab, then choose the **Closed** option from the **Default comment setting for new content** field.
17. Click on **Save and add fields**.
18. On the **Manage fields** page under **Add new field**, enter **Slides** in the **Label** field.
19. Enter **simple_slideshow_slides** in the **Field name** field.
20. Select **Image** from the **Field** drop-down.

21. Select **Image** from the **Widget** field drop-down:

LABEL	NAME	FIELD	WIDGET	OPERATIONS
+ Title	title	Node module element		
+ Body	body	Long text and summary	Text area with a summary	edit delete
+ Add new field				
Slides	field_simple_slideshow	Image	Image	
Label	Field name (a-z, 0-9, _)	Type of data to store.		Form element to edit the data.
+ Add existing field				
	- Select an existing field -		- Select a widget -	
Label	Field to share			Form element to edit the data.

22. Click on **Save**.
23. On the **Field settings** page, leave the options in their default state, and then click on **Save field settings**.
24. On the following page, in the **SIMPLE SLIDE SHOW SETTINGS** section, set the **Minimum image resolution** as **700x300 pixels**.
25. Set the **Number of values** drop-down to **Unlimited**.
26. Click on **Save settings**.
27. Select the **Manage display** tab.
28. In the **Slides** row select **Hidden** from the **Label** drop-down.
29. Select **Slideshow** from the **Format** drop-down.
30. Select the gear at the end of the row.
31. Select the **simple-slideshow** option in the **Image style** field.
32. Check the option **Create prev/next controls**.
33. Click on **Update**, and then on **Save**.

34. Select the **Add content** shortcut from the admin menu:

FIELD	LABEL	FORMAT
Body	<Hidden>	Default
Slides	<Hidden>	Format settings: Slideshow
Image style simple-slideshow		
Link image to Nothing		
Caption Nothing		
Transition effect fade		
Transition speed * 1000 Duration of transition (ms).		
Timeout * 4000 Time between transitions (ms). Enter 0 to disable automatic transitions (then, enable pager and/or controls).		
Order Normal		
<input checked="" type="checkbox"/> Create prev/next controls		
Prev/next controls position After		
<input type="checkbox"/> Pause on hover		
Pager None		
<input type="button" value="Update"/> <input type="button" value="Cancel"/>		

35. Click on **Simple slideshow** and on the **Create Simple slideshow** page enter **Simple carousel** in the **Title** field.

36. Leave the **Body** field empty.

37. In the **Slides section** select **Choose file**, then select and choose a suitable image from your computer.

38. After choosing the file, click on **Upload**.

39. Select **Choose file** again and select and choose another image.

40. Click on **Upload**.

41. Select the **URL path settings** tab and enter **simple-carousel** in the **URL alias** field.
42. Click on **Save**.

Simple carousel

[View](#) [Edit](#)



[Prev](#) [Next](#)

How it works...

We begin by uploading the jQuery cycle plugin to the **libraries** folder. This is a JavaScript library that takes care of the main logic of the slideshow including the slide transitions and the timer functionality.

In the next step, we create a new image format called simple-slideshow. The format of the image format's name must be in lowercase, and not contain spaces. The image format ensures that the slideshow image is output at a resolution of **700x300** pixels.

Now we begin to create a new content type named Simple slideshow. On the **Display settings** tab we uncheck the **Display author and date information** option because the carousel doesn't need date or author information. Similarly, on the **Comment settings** tab we select **Closed**, which prevents comments from being added to the carousel.

After configuring the initial content type settings we proceed to add new fields. We add only one field, an **Image** field, which will be used to store all of the images in the slideshow. After adding the image field we configure its settings. We set the number of values to **unlimited**. This means that the slideshow can have an unlimited number of slides. We also set the minimum resolution to **700x300** pixels. This will enforce that images uploaded to a slide will be at least **700x300** pixels, thus preventing images from being stretched.

After configuring the **Image** field's settings we begin to configure its **Display**. We hide the **Image** label as it is not required for a slideshow, and we set the **Format** to **Slideshow**. It is this setting that will cause the images to display as a slideshow rather than a list of images. Now we select the gear to open the settings for the slideshow. The only change we make is to enable the **Create prev/next controls**.

Creating a document content type

In this recipe, we will demonstrate how to create a simple content type for storing documents or files. This content type forms the basis of the document library, which features in the following recipe.

After completing this recipe, you will be able to upload documents that output content in the following format:

Directions to our office

Submitted by dylan on Fri, 01/27/2012 - 19:14

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis lectus metus, at posuere neque. Sed pharetra nibh eget orci convallis at posuere leo convallis. Sed blandit augue vitae augue scelerisque bibendum. Vivamus sit amet libero turpis, non venenatis urna. In blandit, odio convallis suscipit venenatis, ante ipsum cursus augue.

 Test.docx

How to do it...

We will begin by creating a new content type called Document. We will add a file field to the content type and finally configure its display:

1. Select **Structure** from the admin menu, then select **Content types**.
2. Select **+Add content type**.
3. Enter **Document** in the **Name** field.
4. Click on **Save and add fields**.
5. In the **Add new field** section enter **File** in the **Label** field.
6. In the **Field name** field, enter **document_file**.

Working with Media

7. In the **FIELD** drop-down select **File**:

LABEL	NAME	FIELD	WIDGET	OPERATIONS
⊕ Title	title	Node module element		
⊕ Body	body	Long text and summary	Text area with a summary	edit delete
⊕ Add new field	field_ document_file	File	File	Form element to edit the data.
Label	Field name (a-z, 0-9, _)	Type of data to store.		
⊕ Add existing field	- Select an existing field -		- Select a widget -	Form element to edit the data.
Label	Field to share			

8. Click on **Save**.
9. In the **Field settings** popup leave the options in their default state and click on **Save field settings**.
10. Adjust the **Allowed file extensions** field so that it reads **txt, doc, docx, xls, xlsx, ppt, pptx, pdf**.
11. In the **File directory** field enter **document**.
12. In the **Maximum upload size** field enter **10Mb**.
13. Check the **Enable Description field**:

Allowed file extensions *
txt,doc,docx,xls,xlsx,ppt,pptx, pdf
Separate extensions with a space or comma and do not include the leading dot.

File directory
document
Optional subdirectory within the upload destination where files will be stored. Do not include preceding or trailing slashes.

Maximum upload size
10Mb
Enter a value like "512" (bytes), "80 KB" (kilobytes) or "50 MB" (megabytes) in order to restrict the allowed file size. If left empty the file sizes will be limited only by PHP's maximum post and file upload sizes (current limit **64 MB**).

Enable Description field
The description field allows users to enter a description about the uploaded file.

14. Leave the remaining settings in their default state and click on **Save settings**.
15. Select **MANAGE DISPLAY**.
16. In the **File** row set the **LABEL** drop-down to **<Hidden>**.
17. Ensure that the **Generic file** option is selected for the **FORMAT** column, then click on **Save**.

How it works...

We begin by creating the new Document content type and then adding a new **File** field. The **File** field can be used to store any file type; however, using the **Allowed file extensions** field we restrict the allowable file types to only a small selection. The reason for this is to prevent users uploading files that are potential security risks.

We set the **File directory** to document. This causes all files uploaded to the field to be stored inside a document folder, simply for better organization. The **Maximum upload size** is set to **10Mb**. Regardless of the size setting here, the maximum upload size is limited by the server's PHP configuration.

The **Enable Description** field is checked so that the editor has the opportunity to add a user-friendly link term for the file, instead of the filename.

Finally, we hide the File's label text on the **Manage display** screen, as it's not needed in this case.

See also

In this chapter:

- ▶ *Creating a simple document library*

Creating a simple document library

Many organizations have a requirement for storing and managing documents. This recipe makes use of the Document content type created in the preceding recipe to build a document library, resulting in a document library as depicted in the following screenshot:

Document library	
Document name	Filename
Directions to our office	Test.docx
PDF directions	This is test file 1.

Getting ready

For this recipe you will need to install the most recent recommended releases of the following modules:

- ▶ <http://drupal.org/project/views>
- ▶ <http://drupal.org/project/ctools>

After installing these modules you need to enable the following features:

- ▶ Chaos tools
- ▶ Views
- ▶ Views UI

You will also need to have completed the previous recipe, *Creating a document content type*.

How to do it...

First we will create a new **View** for displaying the document list; we will then configure the view before testing it:

1. Select **Structure** from the admin menu, then select **Views**.
2. Select **+Add new view**.
3. In the **View name** field enter **Document library**.
4. Configure the first fieldset so that it reads **Show Content of type Document sorted by Title**.
5. In the second fieldset, set the **Display format** to **Table**.
6. Leave **Items to display** set as **10**, and leave **Use a pager** checked.

7. Check **Create a menu link** and ensure that **Main menu** is selected in the **Menu** drop-down:

The screenshot shows the 'Create a page' configuration form. At the top, there are dropdown menus for 'Show' (Content), 'of type' (Document), and 'sorted by' (Newest first). Below these are several configuration options:

- Create a page**: A checked checkbox.
- Page title**: A text input field containing 'Document library'.
- Path**: A text input field containing 'http://drupal7cbc7.streetfish.co.uk/document-library'.
- Display format**: A dropdown menu set to 'Table'.
- Items to display**: A text input field containing '10'.
- Use a pager**: A checked checkbox.
- Create a menu link**: A checked checkbox.
- Menu**: A dropdown menu set to 'Main menu'.
- Link text**: A text input field containing 'Document library'.
- Include an RSS feed**: An unchecked checkbox.

8. Click on **Continue & edit**.
9. In the **Fields** section select the link **Content: Title**.
10. In the **Configure** field popup check **Create a label**.
11. Enter **Document name** in the **Label** field.
12. Select **Apply (all displays)**.
13. In the **Fields** section click on the **Add** button.
14. In the **Add fields** popup enter the search term **file** to filter the list of available fields.
15. Check the item named **Content: File – Appears in node:document**.
16. Click on **Apply (all displays)**.
17. In the **Configure** field popup configure the **Label** field to read **Filename**.
18. Click on **Apply (all displays)**.

19. In the **Sort criteria** section select the link **Content: Title (desc)**.
20. In the **Configure sort** criterion popup select **Sort ascending**.
21. Click on **Apply (all displays)**.

The screenshot shows the 'Page details' tab of a View configuration page. The left sidebar contains sections for TITLE, FORMAT, FIELDS, FILTER CRITERIA, SORT CRITERIA, and an ADVANCED button. The main area shows PAGE SETTINGS with Path: /document-library, Menu: Normal: Document library, Access: Permission, and View published content. It also shows HEADER, FOOTER, and PAGER settings.

22. Click on **Save**.
23. Go to Drupal root/document-library to test your new document library; you may need to add some Document nodes if you haven't already.

How it works...

We start by creating a new View, which we will use for displaying the list of documents in the library. We configure the View so that it displays only **Document** nodes, sorted by **Title**, which is effectively the document title.

The **Display format** is set to **Table** because this will enable users to quickly scan through the alphabetical list of document titles.

In the **Fields** section we edit the existing field **Content: Title**, so that it displays a label for the table heading. We then add a new field, **File**, which will output a link to the file. We set the **Label** for the field to **Filename**, which will appear in the table heading for the column.

In the **Sort criteria** section we update the sort order for the **Title** criteria so that the items are output in ascending alphabetical order.

There's more...

So far we have seen how to create a simple document library. However, it doesn't take too many further steps to add some of the commonly requested extra features.

Document categorization

Adding categorization to the document library is relatively easy. First you need to create a new taxonomy to store the list of categories. Then, update the Document content type and add a **Term reference** field that references the categories taxonomy. After updating the Document content type, update the **Document library** view to include a new field to display the category. Finally, add a filter on the **Category** field, setting it to exposed. This will provide the user with a set of checkboxes that can be used for filtering the list of documents.

Linking documents to a content type

We have seen in the previous recipes of this chapter, how to create a document library. This recipe will show you how you can modify the **Basic page content type** so that it contains a lookup field, with which links to documents can be added to the page.

Getting ready

For this recipe you will need to install the most recent recommended releases of the following modules:

- ▶ <http://drupal.org/project/entityreference>
- ▶ <http://drupal.org/project/entity>

You will also need to have completed the previous two recipes in this chapter:

- ▶ *Creating a document content type*
- ▶ *Creating a simple document library*

How to do it...

First we will edit the **Basic page content type**, and add a new **Entity reference** field, which will be used for looking up and storing links to the document library. Following this, we will configure the display of the field so that it is output as a link.

1. Select **Structure** from the admin menu, then select **Content types**.
2. Select **Manage fields** for the **Basic page** content type (or any other suitable content type).
3. In the **Add new field** section, enter **Related documents** in the **Label** field.
4. In the **Field name** field, enter **basic_page_documents**.
5. In the **Field column**, select the option **Entity reference**.
6. In the **Widget** column, select **Autocomplete**:

LABEL	NAME	FIELD	WIDGET	OPERATIONS
⊕ Title	title	Node module element		
⊕ Body	body	Long text and summary	Text area with a summary	edit delete
⊕ Add new field	field_basic_page_doc	Entity Reference	Autocomplete	
Label	Field name (a-z, 0-9, _)	Type of data to store.	Form element to edit the data.	
⊕ Add existing field	- Select an existing field -	- Select a widget -		Form element to edit the data.
Label	Field to share			

7. Click on **Save**.
8. In the **Field settings** popup, select **Node** from the **Target type**.
9. In the **Target bundles** field select **Document**.

10. Leave the **Sort by** field set as **Don't sort**:

FIELD SETTINGS

These settings apply to the *Related documents* field everywhere it is used. These settings impact the way that data is stored in the database and cannot be changed once data has been created.

Target type *
 Node
The entity type that can be referenced thru this field.

Entity selection mode *
 Simple (with optional filter by bundle)

Target bundles
 Advanced slideshow
 Article
 Basic page
 Carousel
 Document
The bundles of the entity type that can be referenced. Optional, leave empty for all bundles.

Sort by
 Don't sort
 A property of the base table of the entity
 A field attached to this entity

11. Click on **Save field settings**.
12. On the following page, in the **RELATED DOCUMENTS FIELD SETTINGS** fieldset, set the **Number of values** field to **Unlimited**.
13. Leave the remaining settings in their default state and click on **Save settings**.
14. Select the **MANAGE DISPLAY** tab.
15. In the **Related documents** row, set the **LABEL** drop-down to **<Hidden>**.
16. Select the gear for the **Related documents** row.
17. Check the option **Link label to the referenced entity**.
18. Click on **Update**, then on **Save**.

19. You will now see a new field, **Related documents**, when editing **Basic page** content where you can look up and link to documents from the document library as shown in the following screenshot:

The screenshot shows a 'Basic page' content editor. At the top, there are 'View' and 'Edit' buttons. The main content area contains a paragraph of placeholder text: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis lectus metus, at posuere neque. Sed pharetra nibh eget orci convallis at posuere leo convallis. Sed blandit augue vitae augue scelerisque bibendum. Vivamus sit amet libero turpis, non venenatis urna. In blandit, odio convallis suscipit venenatis, ante ipsum cursus augue.' Below this text is a section titled 'Related documents:' with two links: 'Directions to our office' and 'PDF directions'.

How it works...

For this recipe we edit the existing **Basic page** content type. Feel free to create a new content type, if you would prefer not to alter the **Basic page** type.

We add a new field to the **Basic page** type, **Entity reference**. This new field type is capable of referencing a range of entities, such as Users, Nodes, and Taxonomy terms. After selecting the **Entity reference** field type, we set the **Widget** to **Autocomplete**. This will provide an autocomplete textfield to lookup the documents. Alternatively, it is possible to use a drop-down instead of an autocomplete field; however, this may quickly become too large to be usable.

On the following page, we configure the reference field so that it is bound to lookup only **Node**. Then, in the **Target bundles** field, we specify that we only want to retrieve nodes of the **Document** content type.

On the following settings page, we set the **Number of values** field to **Unlimited**. This means that the editor can look up an unlimited number of documents to associate with the page.

On the **Manage display** tab for the **Basic page**, we hide the label for the **Related documents** field, as it's not needed. We then select the display configuration gear that allows us to select the option **Link label** to the referenced entity. This means that when the **Basic page** is displayed, the linked **Related document** will output as a link to the document, rather than just plain text displaying its title.

Adding video to a content type

In this recipe we will see how to create a Video content type that has a field for embedding videos. Creating a content type, which can deal specifically with embedded content, is far more powerful than simply pasting a video embed code into the Body field. By having a specific Video field, you can have complete control over the display and positioning of the video, and a greater deal of control over its content and format.

The resulting video display of this recipe is as follows:

Test video

Submitted by dylan on Mon, 01/30/2012 - 19:08

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis lectus metus, at posuere ne
Sed pharetra nibh eget orci convallis at posuere leo convallis. Sed blandit augue vitae augue sceler
bibendum. Vivamus sit amet libero turpis, non venenatis urna. In blandit, odio convallis suscipit
venenatis, ante ipsum cursus augue.

Video:



Getting ready

For this recipe you will need to install the most recent recommended releases of the following modules:

- ▶ <http://drupal.org/project/media>
- ▶ http://drupal.org/project/media_youtube
- ▶ <http://drupal.org/project/ctools>
- ▶ <http://drupal.org/project/views>

You will then need to enable the following features of the above modules:

- ▶ Chaos tools
- ▶ File entity
- ▶ Media
- ▶ Media Internet sources
- ▶ Media: Youtube
- ▶ Views

How to do it...

We will begin by creating a new Video content type to which we will add a **Video** field. We then configure the **Video** field and finally we will see how to add a Video content node:

1. Select **Structure** from the admin menu, then select **Content types**.
2. Select **+Add new content type**.
3. In the **Name** field enter **Video**.
4. Click on **Save and add fields**.
5. In the **Add new fields** section enter **Video** in the **Label** field.
6. In the **Field name** field enter **video**.

7. Select **Multimedia asset** from the **FIELD** drop-down:

LABEL	NAME	FIELD	WIDGET	OPERATIONS
⊕ Title	title	Node module element		
⊕ Body	body	Long text and summary	Text area with a summary	edit delete
⊕				
Add new field				
Video	field_video	Multimedia asset	Media file selector	Form element to edit the data.
Label	Field name (a-z, 0-9, _)	Type of data to store.		
⊕				
Add existing field				
	- Select an existing field -		- Select a widget -	Form element to edit the data.
Label	Field to share			

8. Click on **Save**.
9. On the following page click on **Save field settings**.
10. On the following page, in the section **Allowed remote media types** uncheck the **Image** option and check the **Video** option.
11. In the **Allowed URI schemes** uncheck the **public://** option and check the **youtube://** option.
12. Click on **Save settings**.
13. Select the **Manage display** tab.
14. In the **Video** row select the **Format** drop-down, and choose the **Media** option.
15. Select the gear for the **Video** row.
16. Select **Original** from the **File view mode** drop-down.
17. Click on **Update**, then on **Save**.

FIELD	LABEL	FORMAT	
⊕ Body	<Hidden>	Default	
⊕ Video	Above	Media	File view mode: Original
Hidden			
<i>No field is hidden.</i>			

18. Select the **Add content** shortcut from the admin menu.
19. Select the **Video** content type.
20. In the **Title** field enter **Sample Youtube video**.
21. In the **Description** field enter some text to describe your video, for example, **This is the description text of your video**.
22. Under the **Video** heading, click on the **Select media** button.
23. Select the **web** tab in the **Media select** popup.
24. Search for a Youtube video to embed, and copy its URL or its embed code.
25. Paste the URL or embed code into the **URL or Embed code** field.
26. Click on **Submit**, then on **Save**; you will now see your video content ready to start streaming.

How it works...

We start by creating a new content type, **Video**. We then add a new field, **video**, which we assign the field type, **Multimedia asset**.

On the **Settings** page for the field we remove the option to attach an image to the field, and enable the option to add a video.

We then configure the allowed URI schemes enabling Youtube video embedding, and disabling embedding from the local file system.

After configuring the settings for the Video field we configure the **Display**. We then set the **File view mode** to **Original**, which means that Drupal will output the video with the correct embed code, rather than as an image thumbnail or other type of preview.

After configuring the display, we create a sample **Video** content node. To do this we add a **Title** and **Body** as usual, then we copy and paste a video URL from Youtube to the embed field. The **Media** module automatically takes care of all of the work required to identify what has been entered in the field and renders the video player accordingly.

There's more...

In this recipe, we have seen how to embed Youtube videos into a content node. The method described above is not limited to only Youtube videos, and can be easily expanded.

Embedding videos from other sources

The **Media** module is built in a way that makes it easy to be extended. There are a growing number of modules in the directory, which provide embedding services for sites such as Flickr, Vimeo, and SoundCloud.

You can see all of the Media module plugins that are available by searching the [Drupal.org](#) website using the search term **Media**.

Embedding video directly in a WYSIWYG

It's important not to forget that video such as Youtube can easily be embedded directly in the Body field, or other Long text field. However, this method doesn't provide any of the restrictions of the **Media** field, such as restricting video to a particular source, or setting a limit on the number of embedded videos allowed.

8

Integrating Web APIs

In this chapter we will cover:

- ▶ Integrating with Facebook
- ▶ Displaying a live Twitter feed
- ▶ Adding simple PayPal integration to content types
- ▶ Setting up the Add This social bookmarking service
- ▶ Adding a Google Map to content

Introduction

In this chapter, we will be diving into some of the third-party APIs and services that can be integrated with Drupal to provide a richer user experience.

API stands for Application Programming Interface. Commonly, an API provides a means by which a developer can access the abstracted functions of an application or service to query it and provide some results which can then be used in a third-party application.

In the first recipe, we will be integrating Drupal with Facebook to provide a one-click login service, removing the need for a user to register on your Drupal site. Then, we will see one of the simplest methods of adding a live Twitter feed to a site, without using any third-party modules.

Mid-way through the chapter we have a recipe for integrating Drupal with PayPal. In this recipe we see how to use the PayPal payment button generation service to add "Buy now" buttons to a Product content type.

Near the end of the chapter we have a recipe on adding the *Add this* social bookmarking service to the Drupal site, enabling users to Tweet, Like, or +1 content, amongst other things, with one quick click.

We finish with a recipe on integrating Google Maps into a content type. In this recipe we create a Castle content type, which can be used for storing the details of local castles displaying their geocoded locations on a Google Map.

Integrating with Facebook

The Facebook Connect service allows for a one-click sign on to a Drupal site using the login credentials of the user's Facebook account. In this chapter, we will see how to use the Facebook OAuth module to enable the Facebook Connect authentication service on a Drupal site.

Getting ready

To complete this recipe you will need to have an active Facebook account, and to be logged in. You will also need to install and enable the following module:

- ▶ <http://drupal.org/project/fboauth>

How to do it...

We will start by creating a new Facebook application which will be used to provide the authentication for the domain. We will then configure the Facebook OAuth module with the authentication details provided by the Facebook app. Finally, we will configure the Facebook login module and position it in the header.

1. Go to the following Facebook URL: <https://developers.facebook.com/apps/>.
2. Select **+Create New App**.
3. In the **Create New App** popup enter your **App Display Name**—this could be the name of your site or service.
4. In the **App Namespace**, enter a namespace string consisting of up to **20 lowercase** characters, **hyphens**, or **underscores**—no spaces—this could also be the name of your site, with underscores or hyphens instead of spaces.
5. Check the agree checkbox, then click on **Continue**:

The screenshot shows the 'Create New App' dialog box. It has fields for 'App Display Name' (My authentication app) and 'App Namespace' (my_auth_app). Both fields are marked as valid and available. A checkbox for agreeing to the Facebook Platform Policies is checked. At the bottom are 'Continue' and 'Cancel' buttons.

6. In the **App domain** field enter your domain name—do not enter a sub-domain as the authentication is automatically applied to any sub-domains.
7. In the **Category** field select a category which most relates to your site or service—if in doubt leave it as **Other**.
8. Select the **Website** tab to expand the section and enter the URL to your site in the **Site URL** field.

The screenshot shows the Facebook App Settings page. The 'Basic info' section contains fields for App Display Name (My authentication app), App Namespace (my_auth_app), Contact Email (info@example.com), App Domain (example.com), and Category (Books). The 'Cloud Services' section shows a note about generating a hosting URL. The 'Select how your app integrates with Facebook' section lists several options: Website (selected, Site URL example.com), App on Facebook, Mobile Web, Native iOS App, Native Android App, and Page Tab. A 'Save Changes' button is at the bottom.

9. Click on **Save Changes**.
10. Go to your Drupal site, leaving the Facebook app settings page open in another tab.
11. Select **Configuration** from the admin menu, then select **Facebook OAuth settings**.

12. Go back to the Facebook app page and copy the **App ID**:

My authentication app

App ID: 344290538934747
App Secret: 5be75ceae4e7648a60c4aa4f9551c008 (reset)
⚙ (edit icon)

Basic info

App Display Name: [?] My authentication app

App Namespace: [?] my_auth_app

Contact Email: [?] info@example.com

App Domain: [?] example.com ✖

Category: [?] Books Choose a sub-category

13. Back in Drupal, paste the copied App ID into the **App ID** field of the Facebook OAuth settings page.
14. Go back to the Facebook app page and copy the **App Secret**.
15. Back in Drupal, paste the copied **App Secret** into the **App Secret** field.
16. In the **Basic mapping** section check the option **Import Facebook e-mail address**:

App ID
247786005301869

To use Facebook connect, a Facebook Application must be created. Set up your app in [my apps](#) on Facebook. Enter your App ID here.

App Secret
2d234b1b5103d72573fc99594d26aa9b

To use Facebook connect, a Facebook Application must be created. Set up your app in [my apps](#) on Facebook. Enter your App Secret here.

BASIC MAPPING

Import Facebook e-mail address

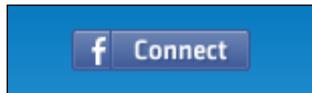
Importing Facebook e-mail addresses requires additional confirmation from the end-user, but it is **strongly recommended** to ensure proper user functionality (password reset, newsletter functionality, etc). If not used, providing a user with some kind of proxy e-mail address is recommended.

User name import

Facebook username (i.e. johnsmith)
 Real name (i.e. John Smith)

Select the Facebook value used to set the user's name when connecting with Facebook.

17. Click on **Save**.
18. Select **Structure** from the admin menu, then select **Blocks**.
19. Find the **Facebook login** block and set its region drop-down to **Header**.
20. Click on **Save blocks**.
21. Go to the homepage; you will now see the Facebook login button in the header:



How it works...

The first step we need to take is to create a **Facebook application**. We will only be using this application for authentication to our site, but it can be used for many more functions, which are beyond the scope of this book.

The **App display** name is the human readable name that will be the title of the app. It's generally recommended to use the name of your website unless you are providing authentication to a specific service, in which case, use the name of that service.

The **App namespace** is a unique identifier for your app, which is used for various purposes such as in the URL to the Facebook application's page. The namespace isn't very important for the purposes of providing authentication.

On the Facebook app's settings page we enter the domain name of the site we're providing authentication for, in the **App domain** field. Only enter your site's domain, as any sub-domains will be covered by this authentication.

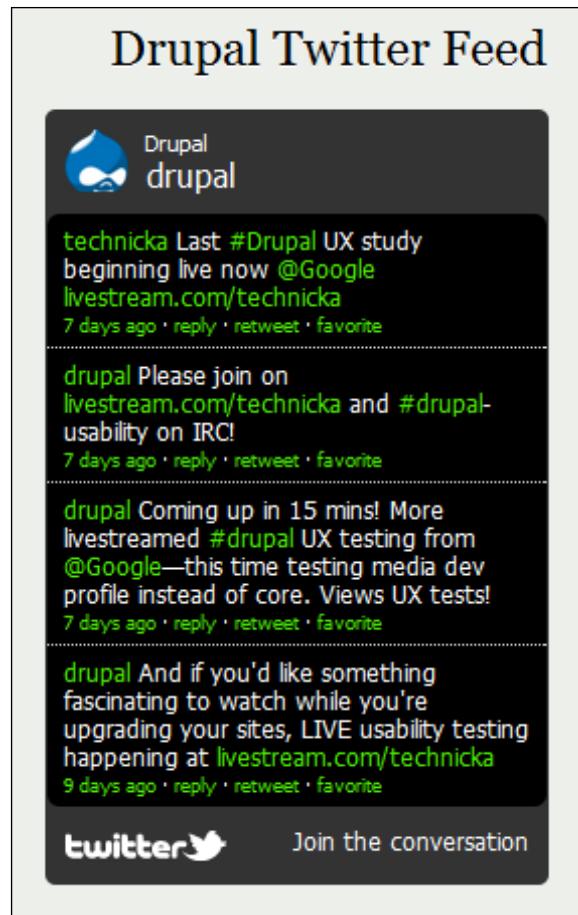
In the **Site URL** field we then enter the URL to the site; this can be a sub-domain or a specific sub-directory of your site, and it is this URL that will be considered the URL of the app.

Back in Drupal we configure the Facebook OAuth settings by entering the **App ID** generated by the Facebook application, and the **App Secret**. These values will be used in the login authentication.

Finally we configure the Facebook login block, which is added by the Facebook OAuth module. We set this to display in the header region.

Displaying a live Twitter feed

Displaying a Twitter feed on a site can help to keep the content fresh, and to show that a product or service is being actively discussed. In this recipe, we will be using a Twitter widget to display the live Twitter feed for Drupal's Twitter account, as follows:



How to do it...

We will first set up a new text format to use for adding JavaScript code. We will then create a new block and generate the Twitter widget code that loads the Twitter feed from Drupal's Twitter account. Finally, we will copy the code and place it in a new block.

1. Select **Configuration** from the admin menu, then select **Text formats**.
2. Select **+Add text format**.
3. In the **Name** field enter **raw**.
4. Ensure that the **administrator** role is checked, and no others.
5. In the **Enabled filters**, do not enable any of the filters.

The screenshot shows the configuration page for a new text format named 'raw'. The 'Name' field is set to 'raw'. The 'Machine name' is also 'raw'. Under the 'Roles' section, the 'administrator' checkbox is checked. The 'Enabled filters' section contains five checkboxes, all of which are unchecked: 'Limit allowed HTML tags', 'Display any HTML as plain text', 'Convert line breaks into HTML (i.e.
 and <p>)', 'Convert URLs into links', and 'Correct faulty and chopped off HTML'.

Name *	raw	Machine name: raw
Roles		
<input type="checkbox"/> anonymous user		
<input type="checkbox"/> authenticated user		
<input checked="" type="checkbox"/> administrator		
Enabled filters		
<input type="checkbox"/> Limit allowed HTML tags		
<input type="checkbox"/> Display any HTML as plain text		
<input type="checkbox"/> Convert line breaks into HTML (i.e. and <p>)		
<input type="checkbox"/> Convert URLs into links		
<input type="checkbox"/> Correct faulty and chopped off HTML		

6. Click on **Save configuration**.

7. Select **Structure** from the admin menu, then select **Blocks**.
8. Select **+Add block**.
9. In the **Block title** field, enter **Drupal Twitter Feed**.
10. In the **Block description** field, enter **Drupal Twitter Feed**.
11. In a new browser tab, go to the URL: <http://twitter.com/about/resources/widgets/>.
12. Select the **My Website** tab.
13. Select **Profile Widget**.
14. In the **Settings** tab, enter **Drupal** in the **Username** field.
15. Select the **Preferences** tab.
16. Check **Yes** for the options **Poll for new results?** and **Include scrollbar?**.

Customize Your Profile Widget

Settings Poll for new results? Include scrollbar?
Preferences Yes Yes
Appearance Behavior
Dimensions Number of Tweets Load all tweets
Dimensions Number of Tweets
4

technicka Last #Drupal UX study beginning live now @Google livestream.com/technicka 7 days ago · reply · retweet · favorite
drupal Please join on livestream.com/technicka and #drupal-usability on IRC! 7 days ago · reply · retweet · favorite
drupal Coming up in 15 mins! More livestreamed #drupal UX testing from @Google—this time testing media dev profile instead of core. Views UX tests! 7 days ago · reply · retweet · favorite
drupal And if you'd like something fascinating to watch while you're upgrading your sites, LIVE usability testing happening at

twitter Join the conversation

17. We will leave the **Appearance** and **Dimensions** tabs with their default configurations and click on **Finish and Grab Code**.
18. Copy the generated code.
19. Back in Drupal, in the **Add block** popup, select **raw** from the **Text format** drop-down.
20. Paste your copied code into the **Block body** field.

21. In the **Region settings** section, select **Triptych first** from the **Bartik** drop-down (or other suitable region if not using the Bartik theme):

The screenshot shows the 'Drupal Twitter Feed' block configuration page. It includes fields for Block title, Block description, Block body (containing a script for a Twitter widget), Text format (set to raw), and Region settings (set to Bartik (default theme) with Triptych first selected). A scroll bar is visible on the right side of the page.

Block title	Drupal Twitter Feed
The title of the block as shown to the user.	
Block description *	Drupal Twitter Feed
A brief description of your block. Used on the Blocks administration page .	
Block body *	<pre><script charset="utf-8" src="http://widgets.twimg.com/j/2/widget.js"></script> <script> new TWTR.Widget({ version: 2, type: 'profile', rpp: 4, interval: 30000, width: 250, height: 300, theme: { shell: { background: '#333333', color: '#ffffff' }, tweets: { </pre>
Text format	raw
More information about text formats ?	
The content of the block as shown to the user.	
REGION SETTINGS	
Specify in which themes and regions this block is displayed.	
Bartik (default theme)	Triptych first

22. Select **Save block**, and select the **Home** link to see the Twitter feed in action.

How it works...

We begin by creating a new **Text format**. We do this because the default text format will cause the Javascript to be cleaned up. So we create a new Text format **raw**, which has no clean up functions. Be sure to only activate this Text format for trusted user roles, as it's possible to run harmful scripts using this method. We only allow the format to be used by **Administrators**.

With our new Text format ready for use, we create a new **Block** to hold the Twitter widget and add a **Block title** and a **Block description**. Next, we go to the Twitter widgets generation page, where we choose to generate a **Profile** widget. This type of widget will display a list of the most-recent tweets from a specified account. There are other types of widgets that you can explore at your leisure.

To configure the widget we enter a username so that the block will only display the tweets from that particular account. In the **Preferences**, we enable the option **Poll for new results**, which will periodically update the list with any new tweets.

We leave the **Appearance** and **Dimensions** tabs with their default settings, but it's important to note that you set custom colors in the **Appearance** tab, and set the width of the widget in the **Dimensions** tab.

After completing the widget's settings, we copy the generated code and go back to our Drupal site. First, we set the **Block** body's **Text format** to **raw**. This is important because it allows the text field to accept JavaScript without applying any automated clean up functions that could remove necessary parts of the code. Once the **Text format** is set, we paste the code into the **Block body** field.

Finally, we set the block to display in the **Triptych first** region, for purely aesthetic reasons, then click on **Save block** to finish.

There's more...

In this recipe we have seen a method of adding a Twitter feed to Drupal, which doesn't require any third-party modules. There are other ways of achieving the same functionality that may be of use to know about.

Displaying a Twitter feed using the Aggregator module

The Aggregator module is shipped with the Drupal core. It allows you to create any number of blocks of RSS feeds that can be set to update at various intervals. The advantage of this method is that it's easy for editors to add their own Twitter blocks. The drawback is that Twitter doesn't publicize its RSS feeds' paths. So you will have to find the path to the required Twitter feed yourself, or use the RSS feed from one of the Twitter re-publication services.

Include the meaning of RSS inside a rectangular box for easy reference.

Third-party Twitter modules

Probably the most user-friendly way of managing Twitter feeds is with a third-party module such as Twitter Profile Widget:

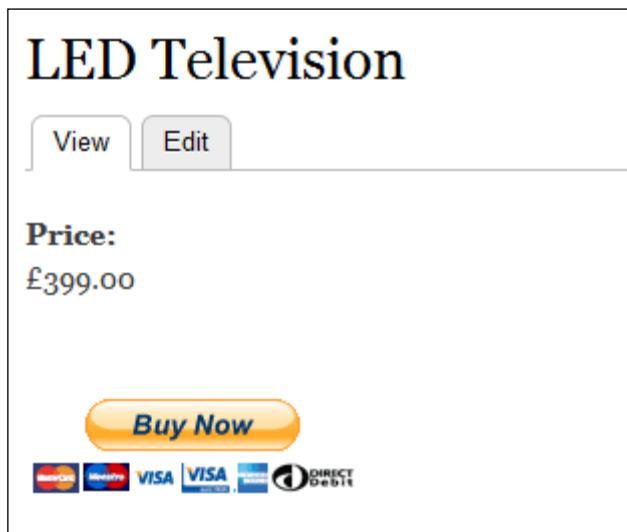
- ▶ http://drupal.org/project/twitter_profile_widget

However, this comes at the cost of adding more module-bulk to your site. Think carefully before adding unnecessary modules that you think the site's editors will need; you may find that they will be comfortable handling small amounts of code.

Adding simple PayPal integration to content types

Creating an online shop can be a daunting task. In this recipe, we will see a method where we add PayPal payments to a content type. We will do this with minimal integration, simply by creating a Product content type with a field to paste generated button code from PayPal.

After completing this recipe, you will have produced a Product node which can be purchased with a **Buy Now** link, as follows:



Getting ready

You will need to have a PayPal business account with **Website Payments Standard** enabled.

How to do it...

In this recipe, we will begin by creating a new Product content type that will have one custom field for storing the PayPal button code. We will then generate a PayPal button for the new product and add it to a new Product node:

1. Select **Structure** from the admin menu, then select **Content types**.
2. Select **+Add content type**.
3. In the **Name** field, enter **Product**.
4. In the **Description** field, enter: **Use the Product content type to display sellable items**.
5. Select the **Display settings** tab and uncheck the option **Display author and date information**.
6. Click on **Save and add fields**.

Name *
Product Machine name: product [Edit]

The human-readable name of this content type. This text will be displayed as part of the list on the *Add new content* page. It is recommended that this name begin with a capital letter and contain only letters, numbers, and spaces. This name must be unique.

Description
Use the Product content type to display sellable items

Describe this content type. The text will be displayed on the *Add new content* page.

7. On the following page, in the **Add new field section**, enter **Price** in the **Label** field.
8. In the **Field name** field, enter **product_price** in the **Label** field and in the **FIELD** column, select **Decimal** from the drop-down.
9. On the following **Settings** page, leave the **Precision**, **Scale**, and **Decimal** fields in their default state and click on **Save field settings**.
10. On the **Product** settings page, check the option **Required field**.
11. Enter **£** in the **Prefix** field.
12. Leave the remaining options in their default state and click on **Save settings**.
13. In the **Add new field section**, enter **PayPal button code** in the **Label** field.
14. In the **Field name** field, enter **paypal_button_code**.
15. Select the **Field type** as **Long text**.
16. Click on **Save**.
17. On the following settings popup click on **Save field settings**.

18. On the **Product settings** page, select the option **Filtered text (user selects text format)** in the **Text processing** field.

19. Click on **Save settings**.

LABEL	NAME	FIELD	WIDGET	OPERATIONS
+ Title	title	Node module element		
+ Body	body	Long text and summary	Text area with a summary	edit delete
+ Price	field_product_price	Decimal	Text field	edit delete
+ Paypal button code	field_paypal_button_code	Long text	Text area (multiple rows)	edit delete
<hr/>				
+ Add new field	field_	- Select a field type -	- Select a widget -	
Label	Field name (a-z, 0-9, _)	Type of data to store.		Form element to edit the data.
<hr/>				
+ Add existing field	- Select an existing field -		- Select a widget -	
Label	Field to share			Form element to edit the data.

20. Back in the **Manage fields** tab, click on **Save**.

21. Select the **Manage display** tab.

22. In the **PayPal button code** row, set the **Label** column to <Hidden>:

FIELD	LABEL	FORMAT	
+ Body	<Hidden>	Default	
+ Price	Above	Default	1 234.12 Display with prefix and suffix. 
+ Paypal button code	<Hidden>	Default	
Hidden			
No field is hidden.			

23. Click on **Save**.

24. Select **Add content** from the admin shortcuts menu.

25. Select the **Product** content type.

26. In the **Title** field enter **LED Television**.

27. Optionally, enter a description of the item in the **Body** field.
28. In the **Price** field, enter **399**.
29. In a separate browser tab, log in to your Paypal account and go to the following URL:
<https://www.paypal-business.co.uk/getting-started-with-payment-buttons/index.htm>
30. Click on **Create button now**.
31. In the step 1fieldset, select **Buy now** from the **Choose button type** field.
32. In the **Item name** field, enter **LED Television**.
33. In the **Item ID** field, enter **P1**.
34. In the **Price** field enter **399**, ensuring **GBP** is selected for the **Currency** field.
35. Leave the customization options in their default state.
36. Enter **25** in the **Postage** field.
37. For the **Merchant account IDs**, leave **Use my secure merchant account ID** selected:

▼ Step 1: Choose a button type and enter your payment details

Choose a button type [Which button should I choose?](#)

Buy Now

Note: [Go to My saved buttons](#) to create a new button similar to an existing one.

Item name Item ID (optional) [What's this?](#)

LED television P1

Price Currency

399 GBP [Need multiple prices?](#)

Customise button Your customer's view

Add dropdown menu with price/option [Example](#)

Add drop-down menu without prices [Example](#)

Add text field [Example](#)

▶ [Customise text or appearance](#) (optional)



Postage

Use specific amount GBP [Help](#)

Merchant account IDs [Learn more](#)

Use my secure merchant account ID

Use my primary email address

38. Click on the **Create** button.
39. On the following page, select and copy the button code.
40. Go back to Drupal, and select **Full HTML** for the **PayPal button code** field.
41. Paste the code into the **PayPal button code** field text area. If the WYSIWYG is enabled for this Text format, go to the source code view before pasting the button code:

The screenshot shows the Drupal content editor interface. At the top left, there is a 'Price' field containing the value '£ 399'. Below it is a 'Paypal button code' field, which contains the following HTML code:

```
<form action="https://www.paypal.com/cgi-bin/webscr" method="post">
<input type="hidden" name="cmd" value="_s-xclick">
<input type="hidden" name="hosted_button_id" value="672QRYCD9B8ES">
<input type="image" src="https://www.paypalobjects.com/en_US/GB/i/btn/btn_buynowCC_LG.gif" border="0" name="submit"
alt="PayPal — The safer way to pay online.">
```

Below the code, there is a 'Text format' dropdown set to 'Full HTML'. To the right, there is a link 'More information about text formats' with a help icon. A note below the text format dropdown states: 'Web page addresses and e-mail addresses turn into links automatically.' and 'Lines and paragraphs break automatically.'

42. Click on **Save**; you will now see the product with the PayPal **Buy now** button.

How it works...

We begin the recipe by creating a new content type, **Product**, which we will use to store sellable items. We create a very simple content type with two extra fields to store the product price and the **PayPal button** code. Of course, it would be very easy to add some more fields, for an image or to add some categorization.

When creating the **Price** field, we set it to be a **Decimal** field type, which is prefixed with a **£** symbol. You can use any other currency symbol here, or none at all.

When creating the PayPal button code field we select the option **Filtered text**, so that when the content is entered, the user can select **Full HTML**. This will ensure that none of the HTML elements of the generated button code are "cleaned up".

Next, we configure the display settings of the content type so it doesn't display the **Label** for the button code field.

Now we begin to add a new product node. First, we create the new node and add a title, and a price. Then we log in to the Paypal account and go to the **Merchant services** page to create a new button. To configure the button we need to add an item name, but we also add an optional **Item ID**, which may be useful for distinguishing between similar products when more products are added. We then add the item **Price**, ensuring that the currency is set to **GBP**. There are a number of **Customisation features** which allow you to add pricing options, and appearance tweaks, which you can explore at your leisure.

We set a specific amount for the **Postage** field, which will be automatically be added to the transaction, and we set the **Merchant account ID** to use the **Secure merchant account ID**. This option increases privacy and security as it doesn't output the e-mail address registered to the Paypal account, and instead uses a secure ID.

After completing the button code setup, we create the button and copy the code into the **Paypal button code** field back in the Drupal site, ensuring to select the **Full HTML Text format** before pasting the code, to prevent un-allowed HTML elements from being removed.

There's more...

We have seen how with just a few clicks we can generate a button to accept payments for an item; however, it's likely that you will want to add more features and complexity.

Expanding on the Paypal button code method

In this recipe, we have seen how Paypal can generate a Pay now button for a product. Paypal provides many more buttons and services that can be used to enrich the user's experience. Paypal makes it easy to add shopping cart functionality to purchase multiple items in one transaction, and also to track stock levels, profits, and redirection pages. All within the Paypal site.

You could then build upon the Product content type by adding a **Category taxonomy**, and create a **Products View** to display lists of products on your site, grouped by category.

Creating a full e-commerce online shop

If you are looking for advanced features such as customer and order management, and complex product pricing rules, then there is an excellent third-party module called Drupal commerce. There are an array of accompanying modules to go with Drupal commerce to add features such as shipping calculators, PDF invoices, and various payment gateways.

See also

- ▶ <https://www.paypal-business.co.uk/getting-started-with-payment-buttons/index.htm>
- ▶ <http://www.drupalcommerce.org/>
- ▶ <http://drupal.org/project/commerce>

Setting up the Add this social bookmarking service

Add this is a sharing platform which takes care of providing links to many of the most popular services that users will want to use to share content. One of the main benefits of the *Add this* service is that it provides useful analytics on the way that content is being shared on your site. The result of this recipe will be a block that displays sharing buttons as follows:



Getting ready

To use the *Add this* bookmarking service, you will need to sign up for an *Add this* account. You can do this by going to the following URL and then selecting **Join now**:

- ▶ <http://www.addthis.com>

How to do it...

We will first set up a new text format to use for adding JavaScript code. We will then create a new block and fetch the *Add this* code, and place it in the block.

1. Select **Configuration** from the admin menu, then select **Text formats**.
2. Select **+Add text format**.
3. In the **Name** field, enter **raw**.
4. Ensure that the **administrator** role is checked, and no others.

5. In the **Enabled filters**, do not enable any of the filters:

Name *
raw Machine name: raw

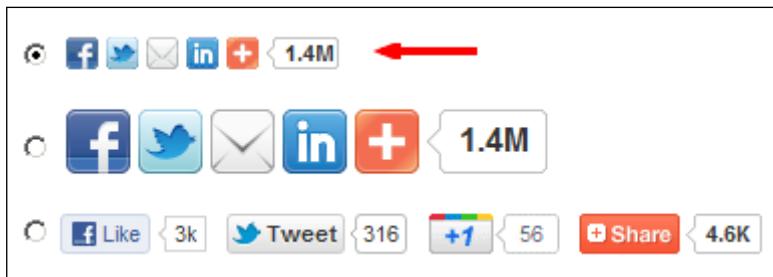
Roles

anonymous user
 authenticated user
 administrator

Enabled filters

Limit allowed HTML tags
 Display any HTML as plain text
 Convert line breaks into HTML (i.e.
 and <p>)
 Convert URLs into links
 Correct faulty and chopped off HTML

6. Click on **Save configuration**.
7. Select **Structure** from the admin menu, then select **Blocks**.
8. Select **+Add block**.
9. Leave the **Block title** field empty.
10. In the **Block description** field, enter **Add this**.
11. In the **Block body**, first select **raw** from the **Text format** drop-down.
12. Go to <http://www.addthis.com> and log in.
13. After logging in, select the first style option in the **Select a style** field:



14. Select **Get the Code**.
15. Copy the generated code and go back to your Drupal site.
16. In the **Block body**, first select **raw** from the **Text format** drop-down.
17. Paste the code into the **Block body** text area.
18. In the **Region settings** section select **Sidebar** first from the **Bartik** field (assuming you are using the Bartik theme, otherwise choose another suitable region).
19. Click on **Save block**.
20. Load the home page of your site and you will now see the *Add this* service rendered in a block in the first column.

How it works...

To start, we need to create a new **Text format**. We do this because the default text format will cause the JavaScript to be cleaned up. So we create a new **Text format raw** that has no clean up functions. Be sure to only activate this **Text format** for trusted user roles, as it's possible to run harmful scripts using this method. We only allow the format to be used by Administrators.

Now that we have created the **Text format** we will need later, the next step is to create a new **Block**. We leave the **Block title** empty as there's no need to tell the users what the block is, as they will very likely be accustomed to using sharing services. After entering an administrative title for the block we need to go and get the code from the *Add this* site. To do this, we log in, and then select which type of buttons to display and copy the code. There are various different types of displays that can be generated, so explore the site at your leisure to see what's available. We then paste the code into the **Body text** area of the site, ensuring that the **raw Text format** has first been selected.

There's more...

It will probably not surprise you that there are a few other ways to accomplish the same or similar functionality as we have seen in the recipe.

Contributed modules

For users who may not want to handle even the smallest snippet of code, there are various contributed modules that will take care of the task of generating social sharing buttons for you.

Template integration

Add this code can be included directly into a `tpl.php` file in the theme, giving extra control over the location where the links are displayed, and under what conditions.

Adding a Google Map to content

Google Maps is a useful tool for describing locations of places, but implementing this manually involves writing some custom JavaScript code, which is not good for non-technical editors. In this recipe, we will be using the Location and GMaps modules to add a Castle content type with a Location field, which outputs its location on a Google Map. The Location field will feature various location attributes and a geocoding service, which will automatically plot the marker on the map, as shown in the following screenshot:

Location:

Carreg cennen castle

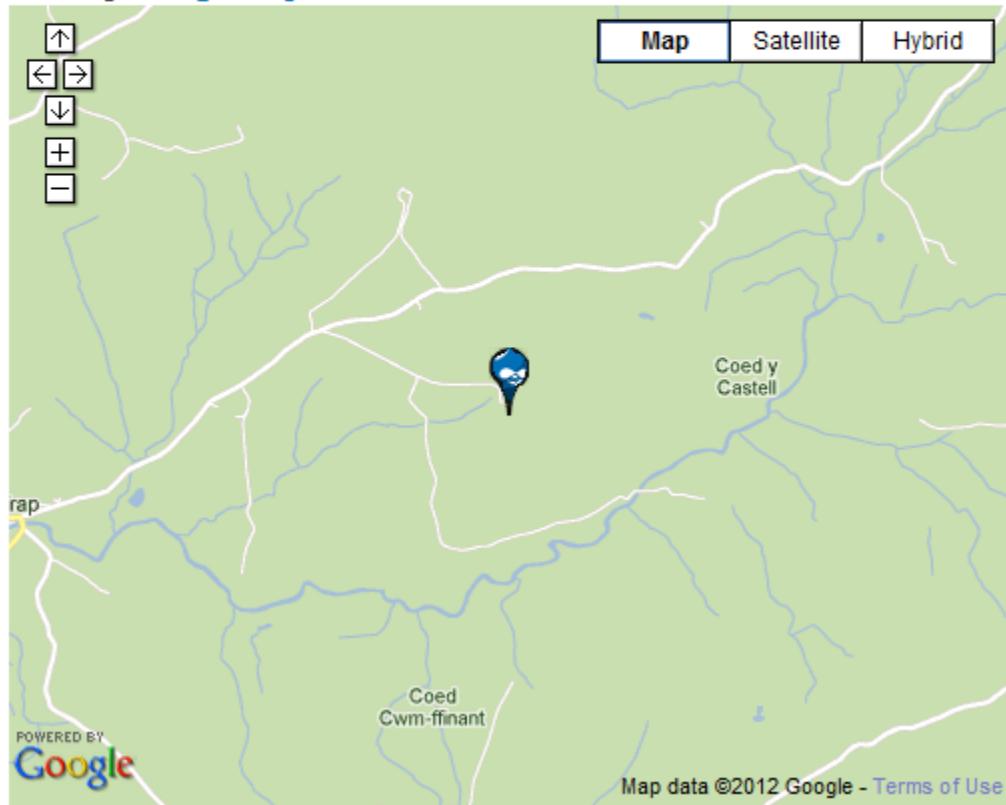
Carreg cennen castle

Llandeilo, Trap, CMN SA19 6UA

United Kingdom

51° 51' 22.4964" N, 3° 56' 11.382" W

See map: [Google Maps](#)



Getting ready

You will need to install the most recent releases of the following modules:

- ▶ <http://drupal.org/project/location>
- ▶ <http://drupal.org/project/gmap>

After installing these modules, enable the following features:

- ▶ Location CCK
- ▶ GMap
- ▶ GMap location
- ▶ Location

How to do it...

In this recipe we will first set up the **Location** and **GMap** modules, and following this we will create a new content type, **Castles**, to which we will add a **Location** field. We will finish the recipe by adding a sample castle to the database.

1. Select **Configuration** from the admin menu, then select **Location**.
2. In the **Location settings** popup, select **United Kingdom** from the **Default country selection** field.
3. Under the **Toggle location display** heading, ensure that the option **Enable the display of locations** is selected.
4. Under the **Province display** heading, ensure that the option **Display province/state code** is selected.
5. Check the option **Use a Google Map to set latitude and longitude**.
6. Leave the remaining options in their default state, then click on **Save configuration**.
7. Select the **Geocoding options** tab.
8. Scroll down the page to find the row for the **United Kingdom**, then select the option **Google maps**.
9. Click on **Save configuration**.
10. Select **Configuration** from the admin menu, then select **Gmap**.

11. Select the link **Google Map API website**, or manually go to the URL:
<http://code.google.com/apis/maps/signup.html>.



The screenshot shows the 'GOOGLE MAP INITIALIZE' page. At the top, there is a 'Google Maps API Key' field containing a long string of characters: ABQIAAAAAnfRLzhF96dMrz_FWHU-b8BTtRtRViNYY_c8-. A red arrow points from this field down to the text 'Your personal Googlemaps API key. You must get this for each separate website at [Google Map API website](#)'. Below this, there is a 'REGENERATE MARKER CACHE' section with a 'Regenerate' button.

12. On the Google maps API signup page, read the terms and conditions, then check the option to agree to them.
13. In the **URL** field, enter your website's domain name, then select **Generate API key**.
14. On the following page, copy the generated API key and go back to your Drupal site.
15. Paste your API key into the field **Google Maps API key**.
16. In the **Default map settings** fieldset, enter **600px** in the **Default width** field.
17. Enter **300px** in the **Default height** field.
18. In the **Map behavior flags** fieldset, check the option **autozoom: Use AutoZoom**.
19. Leave the other options in their default state and click on **Save configuration**.
20. Select **Structure** from the admin menu, then select **Content types**.
21. Select **+Add content type**.
22. In the **Name** field, enter **Castle**.
23. In the **Description** field enter the text: **Use the Castle content type for adding castle information to your site**:



The screenshot shows the 'Add new content type' form. The 'Name' field is filled with 'Castle'. The 'Description' field contains the text 'Use the Castle content type for adding castle information to your site'. Below the description field, there is a larger text area with the placeholder 'Describe this content type. The text will be displayed on the Add new content page.'

24. Click on **Save and add fields**.
25. In the **Add new field section**, enter **Location** in the **Label** field.
26. In the **Field name** field, enter **castle_location**.
27. In the **FIELD** drop-down, select **Location**.

LABEL	NAME	FIELD	WIDGET	OPERATIONS
⊕ Title	title	Node module element		
⊕ Body	body	Long text and summary	Text area with a summary	edit delete
Add new field				
Location	field_castle_location	Location	Location Field	
Label	Field name (a-z, 0-9, _)	Type of data to store.	Form element to edit the data.	
Add existing field				
	- Select an existing field -		- Select a widget -	
Label	Field to share		Form element to edit the data.	

28. Click on **Save**.
29. In the following settings page, in the **Collection settings** fieldset, set the **City** row to **Allow**, in the **COLLECT** column.
30. Set the **State/Province** row to **Allow**.
31. Set the **Postal code** to **Allow**.
32. Leave the other options in their default state, and click on **Save field settings**.

NAME	COLLECT	WIDGET	DEFAULT
⊕ Location name	Allow		e.g. a place of business, venue, meeting point
⊕ Street location	Allow		
⊕ Additional	Allow		
⊕ City	Allow		
⊕ State/Province	Allow	Autocomplete	<input checked="" type="radio"/>
⊕ Postal code	Allow		
⊕ Country	Allow	United Kingdom	<input type="button" value="▼"/>
⊕ Coordinate Chooser	Allow		

33. In the following settings page, leave the options in their default state, and click on **Save settings**.
34. Select **Manage display**.
35. For the **Location** row, set the **Format** to **Address with Map**.
36. Click on **Save**.
37. Select **Add content** from the admin shortcuts menu.
38. Select **Castle**.
39. In the **Title** field, enter **Carreg Cenin**.
40. In the **Description** field, enter **Carreg Cennen Castle is a spectacular Welsh castle lying 4 miles south of Llandeilo**.
41. In the **Location name** field, enter **Carreg Cennen Castle**.
42. In the **Street** field, enter **Carreg Cennen Castle**.
43. In the **City** field, enter **Trapp, Llandeilo**.
44. In the **State/Province** field, enter **Carmarthenshire**.
45. In the **Postal code** field, enter **SA19 6UA**.
46. Ensure **United Kingdom** is selected for the **Country** field.
47. Click on **Save** to finish.

How it works...

We begin by configuring the settings for the **Location** module. The Location module provides the functionality of adding a **Location field** to a content type. We set the **Default country** to the **United Kingdom**; this simply sets the pre-selected country, and can be changed when adding a new item. We set the **Toggle location display** to **Enable the display of locations**. Disabling location display would prevent the address and map from displaying, and is only useful in circumstances where you want to provide a more bespoke output through a custom theme.

We switch on the option to **Use a Google Map** to set the latitude and longitude. This allows the user entering the content to select the point on the map if the geocoding result is not available, or if it is not very accurate.

On the Geocoding options tab we switch on geocoding for the United Kingdom using the Google maps lookup service. Feel free to enable other services for other countries if necessary.

Next, we configure the **GMap module**. The GMap module provides an interface to Google Maps, and is responsible for plotting locations provided by the Location field. To use the GMap module, we must first obtain a Google Maps API key. This provides a form of authentication to the Google Maps service. We obtain the key by going to the Google Maps signup page and entering the URL where the service will be used.

We set the default dimensions of the map display to a more suitable size, and then apply the setting AutoSize. This will cause the map to automatically load at a suitable zoom level. The module doesn't currently allow you to set a custom zoom level.

After configuring the Gmap and Location modules we create a new content type, **Castle**. This is a simple content type to which we add only one custom field, a **Location** field. On the settings popup for the **Location** field, we are presented with a list of location attributes that can be collected at data entry time. We enable the **City**, **State/Province**, and **Postal code** fields as they will be useful attributes to have when describing the location of the castle. Allowing these extra attributes will also improve the accuracy of the geocoding location lookup.

After configuring the **Location** field, we configure the display of the content type in the **Manage display** tab. We set the output of the **Location** field to **Address with map**. Without making this change, no map would be displayed on a Castle node.

Finally, we add a Castle node, entering as many location attributes as possible so that the geocoding service produces a good result. When you look at the final result, you will notice that the term **Carreg Cennen Castle** is output more than once. You can limit which attributes are outputted in the **Display settings** section of the **Location** field's settings page.

9

Creating Regular, Mobile, and Tablet Themes

In this chapter we will cover:

- ▶ Creating a new theme using Zen
- ▶ Overriding HTML output of a content type
- ▶ Creating a "bare-bones" theme from scratch
- ▶ Using the Mobile tools module
- ▶ Installing an off-the-shelf mobile and tablet theme
- ▶ Configuring theme compression and caching

Introduction

We begin this chapter with a recipe on using the highly popular Zen starter theme, which takes much of the pain out of Drupal theme development as it does much of the theme developer's groundwork, allowing you to jump straight in. After seeing how to create a Zen sub-theme we then look at how we can override the HTML output of a content type by creating template files within the theme. We then have a recipe on creating a "bare-bones" theme where we see how to create a very basic Drupal theme completely from scratch.

The proliferation of mobile devices has seen a rise in the demand for mobile-optimized sites. In the recipe, *Using the Mobile tools module*, we will learn how to implement the module that will determine the type of device accessing the site, and respond accordingly by serving an appropriate theme. We then have a recipe on installing an off-the-shelf mobile theme that is optimized for mobile and tablet devices.

We finish this chapter with a recipe describing how to optimize your site for higher performance by configuring caching and theme file aggregation.

Creating a new theme using Zen

In this recipe, we will be creating a sub-theme that inherits from the Zen theme. This is a very popular technique that dramatically reduces the time needed to build a theme as most of the groundwork is already done for you.

Getting ready

For this recipe you will need FTP access to your site and an FTP client if you are not working from localhost.

How to do it...

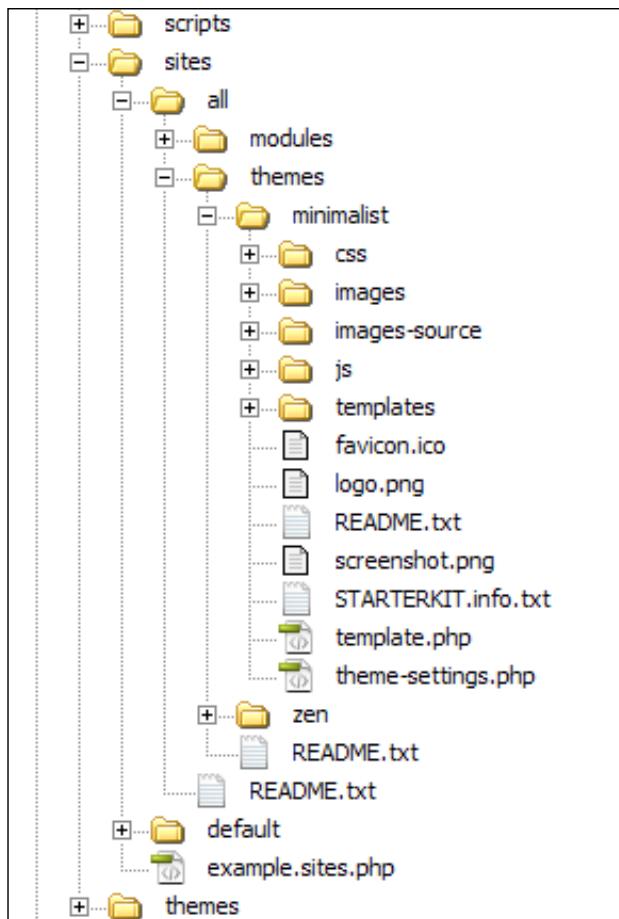
We start by installing and enabling the Zen theme. Once installed, we will create the sub-theme using the STARTERKIT provided by the Zen theme. Finally, we will configure and enable the new theme.

1. Go to the Zen project page: <http://drupal.org/project/zen>.
2. Copy the tar.gz URL of the latest **Recommended release** of the Zen theme:

Downloads				
Recommended releases				
Version	Downloads	Date	Links	
7.x-3.1	tar.gz (212.75 KB) zip (239.92 KB)	2011-Apr-25	Notes	
6.x-2.1	tar.gz (175.83 KB) zip (226.03 KB)	2011-Apr-05	Notes	
5.x-1.2	tar.gz (317.17 KB) zip (332.77 KB)	2009-Feb-15	Notes	

3. Go to your Drupal site and select **Module** from the admin menu.
4. Select **+Install new module**.
5. Paste the URL in the **Install from URL** field, then click on **Install**.
6. Select **Enable newly added themes**.

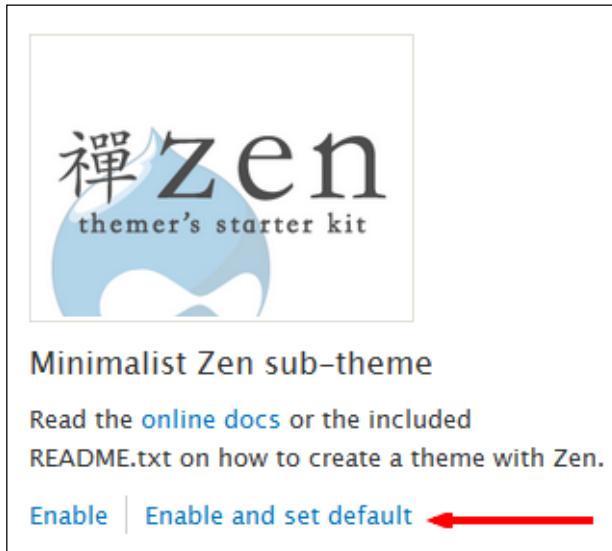
7. Scroll to the **Disabled themes** section and select **Enable** for the newly installed Zen theme.
8. In your FTP client, go to Drupal root/sites/all/themes/zen.
9. Copy the folder STARTERKIT to the folder Drupal root/sites/all/themes/.
10. Rename the STARTERKIT folder to minimalist, and the copied theme folder should now have the path Drupal root/sites/all/themes/minimalist:



11. In the new minimalist theme's folder, rename STARTERKIT.info.txt to minimalist.info.
12. Open minimalist.info and edit the name field to read as follows:

```
name = Minimalist Zen sub-theme
```
13. Back in Drupal, select **Configuration** from the admin menu, then select **Performance**.

14. Select **Clear all caches**.
15. Select **Appearance** from the admin menu.
16. Find the new sub-theme **Minimalist Zen sub-theme**, then select **Enable and set default**:



17. Go to your home page and you will now see the minimalist theme in use.

How it works...

We begin by installing the Zen theme using the installer. We then enable the theme, but we don't set it as the default theme. This is because the new sub-theme we are going to create will be the default theme.

The Zen theme is a starter theme. It consists of a base theme, Zen, and a sub-theme prototype, which is used to create the sub-theme. The sub-theme inherits CSS and custom markup templates from the Zen theme. This means that many of the standard HTML and CSS features of the theme are already taken care of in the parent theme, allowing you, as a theme developer, to quickly create a standards compliant sub-theme without having to start from scratch.



It's important to never edit anything within the Zen theme's folder, because the theme is likely to be updated, and changes may be lost.

After installing the Zen theme, we see that there is a folder within the Zen theme called STARTERTHEME. This folder should only be copied and never edited directly, as it is the prototype for a new sub-theme. We copy the STARTERTHEME folder to the sites/all/themes/ folder and rename it to minimalist. After renaming the folder, we rename the info file to minimalist.info.

The info file contains metadata about the theme such as its name, description, version, and parent theme. In addition to the Name attribute, the info file must contain the Core attribute, which identifies which version of Drupal the theme is compatible with.

The info file defines which regions are available to the theme, in addition to listing the JavaScript and CSS files used in the theme, and also the paths to these files. Take a look at the [Drupal.org](https://www.drupal.org) documentation of the info file for more info on the optional data attributes that can be used.

We edit our renamed info file (`minimalist.info`) and the only attribute we need to change in this case is the Name.



The theme's folder name and the info file must be the same.



Finally, we clear the cache and then enable the new sub-theme. The sub-theme is now ready to start theme development.



After making any changes to the info file, the theme cache needs to be cleared to ensure that the system recognizes the theme. The cache also needs to be cleared each time the .info file is updated.



There's more...

Now let's talk about some other options, or possibly some pieces of general information that are relevant to this task.

Theme inheritance

In this recipe, we have created a sub-theme from the Zen parent theme. It is possible to create a further sub-theme from the sub-theme. For example, you might have a regular theme that has been developed as a sub-theme of the Zen theme. Then, for the Christmas period you may want to simply change the header and maybe some colors of the theme, but rather than editing the regular theme, you can create a further sub-theme which simply specifies a different header and some different color values, leaving the regular theme completely intact.

Theme override functions

In a theme's `template.php` file, it is possible to specify override functions to modify or preprocess the markup, or the variables, before they are output to the theme. For example, you can add a function to customize the output of the breadcrumbs. The advantage of these functions is that they are neatly encapsulated away from the markup of the templates.

See also

- ▶ The Zen theme, at <http://drupal.org/node/193318>
- ▶ The structure of the `.info` file, at <http://drupal.org/node/171205>
- ▶ How to build your own sub-theme, at <http://drupal.org/node/1010576>

Overriding HTML output of a content type

For many purposes, creating a content type and re-ordering the fields is enough. However, for more custom implementations you will need to modify the HTML that is output by the system. Drupal provides an elegant mechanism for overriding HTML output using template files, which specify custom HTML markup, encapsulated as a property of the theme.

Getting ready

We will be using the minimalist theme created in the first recipe of this chapter, *Creating a new theme using Zen*. Please complete this recipe before continuing, and ensure that you have the minimalist theme enabled.

We are going to be editing the theme, so you will need FTP access to the folder unless you are working from your localhost.

How to do it...

In this recipe, we will be creating a **Recipe** content type and creating a sample **Recipe** node. We will then create a node template to which we will add some custom markup to apply to the Recipe node. We will then apply some CSS styles to the Recipe node and finally we will override the HTML output of the ingredients field.

1. Select **Structure** from the admin menu, then select **Content types**.
2. Select **+Add content type**.
3. In the **Name** field, enter **Recipe**.

4. In the **Description** field, enter **Use the Recipe content type to add recipes to the site**:

Name *	Recipe	Machine name: recipe [Edit]
<p>The human-readable name of this content type. This text will be displayed as part of the list on the <i>Add new content</i> page. It is recommended that this name begin with a capital letter and contain only letters, numbers, and spaces. This name must be unique.</p>		
Description		
Use the Recipe content type to add recipes to the site		
<p>Describe this content type. The text will be displayed on the <i>Add new content</i> page.</p>		

5. Click on **Save and add fields**.
6. In the **Add new field** section, enter **Ingredients** in the **Label** field.
7. Enter **recipe_ingredients** in the **Field name** field.
8. In the **FIELD** column, select **Text**.
9. Click on **Save**.

LABEL	NAME	FIELD	WIDGET	OPERATIONS
⊕ Title	title	Node module element		
⊕ Body	body	Long text and summary	Text area with a summary	edit delete
Add new field <input type="text" value="Ingredients"/> Label <input type="text" value="field_recipe_ingredients"/> Field name (a-z, 0-9, _) <input type="text" value="Text"/> Type of data to store. <input type="text" value="Text field"/> Widget Form element to edit the data.				
Add existing field <input type="text"/> Label <input type="text" value="- Select an existing field -"/> Field to share <input type="text" value="- Select a widget -"/> Widget Form element to edit the data.				

10. On the **FIELD SETTINGS** popup, leave the **Maximum length** field set with its default value **255**.

11. Click on **Save field settings**.

12. On the following settings popup, ensure that the **Text processing** field is set to **Plain text**.

13. Set the **Number of values** field to **Unlimited**.

14. Click on **Save settings**.

15. Select **Add content** from the admin shortcuts menu.

16. Click on **Recipe** content type and in the **Title** field, enter **Spanish omelette**.

17. Set the **Text format** of the **Body** field to **Filtered HTML**.

18. Add the following text to the **Body** field:

```
<ol>
- Switch on the grill to a medium heat
- Chop an onion and a chilli then add to an omelette pan and begin to fry with a little olive oil
- In a mixing bowl, beat three eggs with a splash of milk, a pinch of salt and some black pepper
- When the onions are golden brown, add the egg mixture to the pan
- Cook for two minutes
- Place the pan under the grill for ten minutes, or until golden brown

</ol>
```

19. In the **Ingredients** field enter the following ingredients, selecting **Add another** item after entering each ingredient to add a new field:

- 3 eggs**
- Salt**
- Black pepper**
- Milk**
- Olive oil**
- 1 medium sized onion**
- 1 chilli**

Title *
Spicy Spanish omelette

Body (Edit summary)

```
<ol>
<li>Switch on the grill to a medium head</li>
<li>Chop an onion and a chilli then add to an omelette pan and begin to fry with a little olive oil</li>
<li>In a mixing bowl, beat three eggs with a splash of milk, a pinch of salt and some black pepper</li>
<li>When the onions are golden brown, add the egg mixture to the pan</li>
<li>Cook for two minutes</li>
<li>Place the pan under the grill for ten minutes, or until golden brown</li>
</ol>
```

Text format Filtered HTML [More information about text formats](#)

- Web page addresses and e-mail addresses turn into links automatically.
- Allowed HTML tags: <a> <cite> <blockquote> <code> <dl> <dt> <dd>
- Lines and paragraphs break automatically.

INGREDIENTS:

+ 3 eggs
+ Salt
+ Black pepper
+ Milk
+ Olive oil
+ 1 medium size onion
+ 1 Chilli

Add another item

20. Click on **Save**.

[View](#) [Edit](#)

Submitted by [dylan](#) on Wed, 02/22/2012 - 19:08

1. Switch on the grill to a medium head
2. Chop an onion and a chilli then add to an omelette pan and begin to fry with a little olive oil
3. In a mixing bowl, beat three eggs with a splash of milk, a pinch of salt and some black pepper
4. When the onions are golden brown, add the egg mixture to the pan
5. Cook for two minutes
6. Place the pan under the grill for ten minutes, or until golden brown

Ingredients:

3 eggs
Salt
Black pepper
Milk
Olive oil
1 medium size onion
1 Chilli

21. In your FTP client, copy the file `Drupal root/modules/node/node.tpl.php` to the `templates` folder of your minimalist theme, `Drupal root/sites/all/themes/minimalist/templates/`.
22. Rename the copied file to `node--recipe.tpl.php`. Its full path should now be `Drupal root/sites/all/themes/minimalist/templates/node--recipe.tpl.php`.
23. Open the new file `node--recipe.tpl.php`.
24. Update the content of the file as follows:

```
<div id="node-<?php print $node->nid; ?>" class="<?php print
$classes; ?> clearfix"<?php print $attributes; ?>>

<?php print $user_picture; ?>

<?php print render($title_prefix); ?>
<?php if (!empty($page)): ?>
    <h2<?php print $title_attributes; ?>><a href="<?php print
$node_url; ?>"><?php print $title; ?></a></h2>
<?php endif; ?>
<?php print render($title_suffix); ?>

<?php if ($display_submitted): ?>
    <div class="submitted">
        <?php print $submitted; ?>
    </div>
<?php endif; ?>

<div class="content clearfix"<?php print $content_attributes;
?>>
    <?php
        // We hide the comments and links now so that we can render
        them later.
        hide($content['comments']);
        hide($content['links']);
    ?><div class="recipe-instructions"><?php print
render($content['body']);?></div><?php
    ?><div class="recipe-ingredients"><?php print
render($content['field_recipe_ingredients']);?></div><?php
print render($content);
    ?>
</div>

<?php print render($content['links']); ?>
<?php print render($content['comments']); ?>

</div>
```

25. Save and upload the file.
26. Open the following CSS file: Drupal root/sites/all/themes/minimalist/css/nodes.css.

27. Add the following lines to the bottom of the nodes.css file:

```
.node-recipe .recipe-instructions {float:left; width:550px}
.node-recipe .recipe-ingredients {float:left; width:190px;
background:#CCCDAF; padding:10px}
```

28. Save and upload the nodes.css file.

29. In your FTP client, copy the file Drupal root/modules/field/theme/field.tpl.php to the templates folder of your minimalist theme, Drupal root/sites/all/themes/minimalist/templates/.

30. Rename the copied file to field-field_recipe_ingredients.tpl.php. Its full path should now be Drupal root/sites/all/themes/minimalist/templates/field-field_recipe_ingredients.tpl.php.

31. Open the newly copied file field-field_recipe_ingredients.tpl.php.

32. Remove the following comment from the file:

```
<!--
THIS FILE IS NOT USED AND IS HERE AS A STARTING POINT FOR
CUSTOMIZATION ONLY.
See http://api.drupal.org/api/function/theme_field/7 for details.
After copying this file to your theme's folder and customizing it,
remove this
HTML comment.
-->
```

33. Update the content of the file as follows:

```
<div class=<?php print $classes; ?> clearfix<?php print
$attributes; ?>>
<?php if (!$label_hidden): ?>
    <div class="field-label"<?php print $title_attributes;
?>><?php print $label ?>:&nbsp;</div>
    <?php endif; ?>
    <ul class="field-items"<?php print $content_attributes; ?>>
        <?php foreach ($items as $delta => $item): ?>
            <li class="field-item <?php print $delta % 2 ? 'odd' :
'even'; ?>"<?php print $item_attributes[$delta]; ?>><?php print
render($item); ?></li>
        <?php endforeach; ?>
    </ul>
</div>
```

34. Save and upload the file.
35. Clear your cache by selecting **Configuration** from the admin menu, then selecting **Development**, and finally selecting the button **Clear all caches**.
36. Go to the newly created recipe; you will see a bulleted list of ingredients on the right-hand side of the page:

Spicy Spanish omelette

[View](#) [Edit](#)

Submitted by [dylan](#) on Wed, 02/22/2012 - 19:08

1. Switch on the grill to a medium heat
2. Chop an onion and a chilli then add to an omelette pan and begin to fry with a little olive oil
3. In a mixing bowl, beat three eggs with a splash of milk, a pinch of salt and some black pepper
4. When the onions are golden brown, add the egg mixture to the pan
5. Cook for two minutes
6. Place the pan under the grill for ten minutes, or until golden brown

Ingredients:

- 3 eggs
- Salt
- Black pepper
- Milk
- Olive oil
- 1 medium size onion
- 1 Chilli

How it works...

To begin, we create a new content type that we can customize. We create a new **Recipe** content type to which we add one custom text field to store the ingredients. We configure the text field so that its **Number of values** is **Unlimited**, so we can add an unlimited number of ingredients.

After creating the **Recipe** content type we create a new **Recipe** node, which we will use in the templating process.



When adding content in step 18, you might find it easier to use the WYSIWYG editor for adding bulleted list content.

After saving the content, you will see the recipe for Spicy Spanish omelette with instructions on how to make it, with the ingredients list below. What we will do next is create a custom node template for the Recipe node where we will override the output so that the instructions are output inside a `div` with the class `recipe-instructions`, and the ingredients are output inside a `div` with the class `recipe-ingredients`.

To create our node template we copy the `node.tpl.php` file from Drupal's `modules/node` folder into the templates folder of our theme. If the file is kept with the filename `node.tpl.php`, then all nodes throughout the system will be output according to the template defined within this file. Without modifying the `node.tpl.php` file, the output will be identical to the output when a template is not used.

In this recipe, we don't want to target all nodes throughout the system; we only want to create a template for **Recipe** nodes. To do this, we rename the file to `node--recipe.tpl.php`. The first part of the filename identifies the template as a node template; the second part identifies it more specifically as a **Recipe** node template. Note that the second part of the filename, `recipe`, needs to match the machine name of the content type exactly. Also note the double dash `--`, which separates the two terms in the filename. Two dashes are used to separate terms, so that one dash can still be used for spacing words in single terms; for example, if we had a content type called **recipe-book**, we could separate the terms in the filename as follows: `node--recipe-book.tpl.php`.

When editing the new template file `node--recipe.tpl.php`, the first change we make is to add the CSS class `clearfix` to the content div's list of CSS classes. We do this so that the divs after the content block, for example, the comments div, are not floated up with the divs inside the content block.

We then add these two lines:

```
?><div class="recipe-instructions"><?php print  
render($content['body']);?></div><?php  
  
?><div class="recipe-ingredients"><?php print render($content  
['field_recipe_ingredients']);?></div><?php
```

In the first line, we are outputting the body content, with a div wrapper that has the CSS class `recipe-instructions`. In the following line, we are outputting the ingredients field, inside a div wrapper that has the CSS class `recipe-ingredients`.

The important part to note is that we are rendering specific fields from the `$content` variable. We output these fields using the `render` function to which we supply the `$content` variable and the field name key, exactly as they appear on the Manage fields page for the content type.

The following line in the template, which has not been altered, is:

```
print render($content);
```

This statement will output all of the remaining fields configured to be outputted for the content type.



Drupal will only output fields that have not already been output, which is why we do not see the recipe instructions and recipe ingredient being output twice.



So far, the output will now display the instructions and ingredients fields in their own divs. We now need to apply some CSS. To do this we edit the template's `node.css` file. There are numerous CSS files in a Zen template, with the intention of keeping similar CSS instructions encapsulated in different files. We choose to add our new CSS instructions to the `node.css` file because we are affecting the display of a type of node.

We add only two lines to the CSS file:

```
.node-recipe .recipe-instructions {float:left; width:550px}  
.node-recipe .recipe-ingredients {float:left; width:190px;  
background:#CCCDAF; padding:10px}
```

On the first line we target the class `node-recipe`, and then any child divs with the class `.recipe-instructions`. We apply the style `float:left`, causing the instructions box to go to the left, and we specify `width` of `550px`, which leaves enough room for a reasonable sized box to the right.

On the second line we target the `recipe-ingredients` div, also configuring it to float to the left, and this time with a smaller `width` of `190px`. We also give the box a background color and some padding.



It's good practice to be as *specific* as possible when defining your CSS selectors in Drupal, as the CSS classes often appear in different content displays, views and blocks. For example, the class we have used above, `.node-recipe`, will target all appearances of the **Recipe** node, but you may only want to target the node when its output inside a block, or on the homepage.



Now we have applied styling to the **Recipe** node, the node will display with the instructions div to the left, with the ingredients div on the right. However, to further improve the appearance of the instruction box configure the ingredients to be output in an unordered list.

To do this we need to apply a field template to the ingredients field. First we copy the field template prototype `field.tpl.php` from the Drupal folder `modules/field/theme` into our theme's template folder. We then rename the file so that it targets only fields with the machine name `field_recipe_ingredients`, exactly as listed in the **Manage fields** page for the content type, resulting in the filename `field--field_recipe_ingredients.tpl.php`.



For more specificity we could add the content type's machine name, after the field name. This may be useful where the field is being used in multiple content types.

When editing the field template, we change the field-items `<div>` to a `` and update the corresponding closing `</div>` to a ``. We then change the field-item `<div>` to a ``, and also update the corresponding closing `</div>` to a ``. This has the effect of outputting the field items as `` tags instead of `<div>` tags, and therefore outputting them as an unordered list.



When testing theme templates, it's common to find that the template doesn't seem to be working. If you find yourself in this situation, check and re-check the filenames, and also clear the Drupal cache.

See also

- ▶ Drupal 7 theme hook suggestions, at <http://drupal.org/node/1089656>

Creating a "bare-bones" theme from scratch

The benefits of using a "starter" theme should not be under-estimated as they provide a means of rapidly producing a standards-compliant, cross-browser theme, where the groundwork has been done before you begin. However, it's important for a theme developer to understand the architecture of a Drupal theme from the ground up.

This recipe is not so much about what features and functionalities a Drupal theme *can* contain, but instead, what they *should* contain, as a bare minimum.

In this recipe, we will be building a very basic theme which will contain no CSS styling, HTML resets, or custom regions, but only the bare framework for "themers" who want to develop a theme in its entirety.

Getting ready

For this recipe you will need FTP access to your site and an FTP client if you are not working from localhost.

How to do it...

We will start the recipe by creating a blank folder for the new theme to which we add the info file. We will then configure the info file so that it has the required attributes. Then we create the CSS and JavaScript files defined in the info file, and upload the theme to the site.

1. Working locally, create a new folder called `skeleton`.
2. Inside the `skeleton` folder, create a file called `skeleton.info`.
3. Open the file `skeleton.info` and add the following content:

```
name = Skeleton theme
description = A simple 'bare bones' skeleton theme for Drupal
core = 7.x

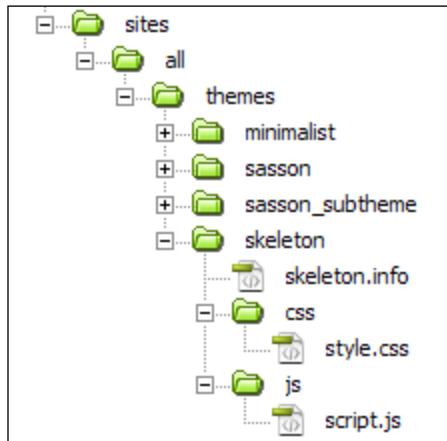
; regions
regions[header] = Header
regions[highlighted] = Highlighted
regions[help] = Help
regions[content] = Content
regions[sidebar_first] = Left sidebar
regions[sidebar_second] = Right sidebar
regions[footer] = Footer

; stylesheets
stylesheets[all][] = css/style.css

; javascript
scripts[] = js/script.js
```

4. Save and close the file.
5. Inside the `skeleton` folder, create a new folder named `css`.
6. Inside the `css` folder, create a new empty file named `style.css`.
7. Inside the `skeleton` folder, create a new folder named `js`.
8. Inside the `js` folder, create a new empty file named `script.js`.

9. Upload your skeleton directory to your site at the location Drupal root/sites/all/themes/:



10. Select **Configuration** then select **Performance**.

11. Select **Clear all caches**.

How it works...

We begin by creating a new folder for the new theme in which we create the skeleton info file, `skeleton.info`. The info file is all that is required for a theme to be recognized as a theme.



The filename of the info file is important as it forms the machine name of the theme.

We add the required attributes to the info file, starting with the name:

```
name = Skeleton theme
```

This is the human readable name of the theme:

```
description = A simple 'bare bones' skeleton theme for Drupal
```

The description is not required, but it's good practice to add a short piece of text to help identify the theme. This text is displayed on the **Appearances** page:

```
core = 7.x
```

The core property specifies which version of the Drupal core the theme is intended to work with. This property is required:

```
; regions  
regions[header] = Header  
regions[highlighted] = Highlighted  
regions[help] = Help  
regions[content] = Content  
regions[sidebar_first] = Left sidebar  
regions[sidebar_second] = Right sidebar  
regions[footer] = Footer
```

The regions section of the info file defines the regions that are available to the theme. These are the standard regions, but you are free to define custom regions, which you can then implement in a TPL file.



If no regions are defined in the info file, then the above regions will be assumed. However, if you should then include one or more custom regions, the default regions will not be available in your theme. This may cause some blocks to become disabled, as they no longer have a region to be applied to.

```
; stylesheets  
stylesheets[all][] = css/style.css
```

In the stylesheets section we define one stylesheet for the theme that is located in the CSS folder of the theme. It is this definition that causes Drupal to load the stylesheet into the header of the page. We can define multiple stylesheets here if necessary.

```
; javascript  
scripts[] = js/script.js
```

Similarly, we define one JavaScript file that is located in the js folder of the theme.

After saving and closing the info file, we go about creating the js and css folders, and the files to go in them. We create one blank CSS file and one blank JavaScript file merely as placeholders for when you are ready to begin the theming process.

Finally, we upload the theme folder to the themes folder on our site. We then clear the site cache to ensure that the theme is recognized by the system. Checking the **Appearances** page you will notice that the new theme is now available.

There's more...

In this recipe, we have built a very basic theme to illustrate the minimum that needs to be done to create a theme from scratch. You will probably want to add a few more features to your theme, which will improve its quality.

Adding a favicon

Almost all sites today use favicons. Implementing a favicon in a Drupal theme is easy; simply add the `favicon.ico` file to the root of your theme and Drupal will do the rest.

Adding a theme screenshot

The theme screenshot is visible on the **Appearances** page. It helps to identify the theme when there are multiple themes installed, and it also helps to create a polished image for the theme.

To add a screenshot for a theme, add the image file named `screenshot.png` to the root of the theme.

Go to the following link to see the best practices for making a Drupal theme screenshot:

- ▶ <http://drupal.org/node/647754>

See also

- ▶ Theming Drupal 6 and 7, at <http://drupal.org/theme-guide/6-7>
- ▶ Structure of the `.info` file, at <http://drupal.org/node/171205>

Using the Mobile tools module

In this recipe, we will be using the Mobile tools module to bring mobile device detection to our site. Using this module we will configure the site to display a different theme on detection of a mobile browser.

Getting ready

To complete this recipe you will need to have installed the following module, enabling only the Mobile tools feature:

- ▶ http://drupal.org/project/mobile_tools

In order to test the results of this recipe you will need to either have a smartphone, or a smartphone emulator.

How to do it...

We will begin by configuring the Mobile tools module. We will then assign a theme to use when a mobile device has been detected before testing the result.

1. Select **Configuration** from the admin menu, then select **Mobile tools**.
2. In the **General configuration** fieldset, enter the URL of your site in the **Mobile URL** field, so that the **Mobile URL** field has the same value as the **Desktop URL** field:

GENERAL CONFIGURATION

Enter the mobile and desktop url for your site. If both urls are equal there will be no redirection, but only theme switching. Go to "theme switching" to configure the theme

Mobile URL

Give the name of your mobile site. It is recommended to use the convention of m.domain .com or www.domain.mobi

Desktop URL

Give the name of your regular website.

3. Click on **Save configuration**.
4. Select the **Theme switching** tab.
5. Under the heading **When do you want to switch theme**, select the option **Switch theme for a mobile device**.
6. In the **Mobile theme** field, select **Bartik**, or preferably any theme other than your default theme.

THEMING CONFIGURATION

You can assign a variation of your current theme to all mobile users . this allows you to configure your theme specific for mobile users. See [help](#) for more information on this configuration. In order to use this functionality you will have to manually create a second *.info file in your theme directory.

When do you want to switch themes

- No theme switch
 Switch theme for a mobile device *
 Switch theme based on the URL

7. Click on **Save configuration**.
8. Load your site in your mobile browser and you will now see the site displayed using the theme selected in step 6.

How it works...

We begin by configuring the **Mobile URL** to make it the same as the **Desktop URL**. This means that there will be no redirection, so the mobile version of the site will be just a different theme.

In the **Theme switching** tab we select the option to **Switch theme for a mobile device**. This means that when the module detects that the site is being viewed by a mobile device, it will switch the theme. We then set the theme that it switches to.

When you load your site in a mobile browser, the site will now be displayed in the theme selected above.

There's more...

In this example we have seen how easy it is to implement device detection and theme switching. However, there is another approach to the mobile website which we have not covered.

Using a different sub-domain or directory for the mobile site

The alternative to theme switching is using a different directory or sub-domain for the site. That is to say, using a separate site altogether. This is a flexible option that gives a higher degree of control over the content that is displayed on a mobile device, and also the layout. The disadvantage is that each item of content has to be posted twice.

Using a device-specific theme

In the **Theme switching** tab of the Mobile tools module configuration, there are options to specify a different theme for the following devices:

- ▶ iPhone
- ▶ iPad
- ▶ iPod
- ▶ Android
- ▶ Blackberry

This feature may become very useful for solving device-specific style issues.

Installing an off-the-shelf mobile and tablet theme

In this recipe, we are going to be installing and configuring the Sasson base theme, which we will use in conjunction with the Mobile tools module to render a mobile theme for mobile and tablet devices.

Getting ready

We will be using the Mobile tools module to display this theme, so please ensure that you have completed the previous recipe, *Using the Mobile tools module*.

For this recipe you will need FTP access to your site and an FTP client.

How to do it...

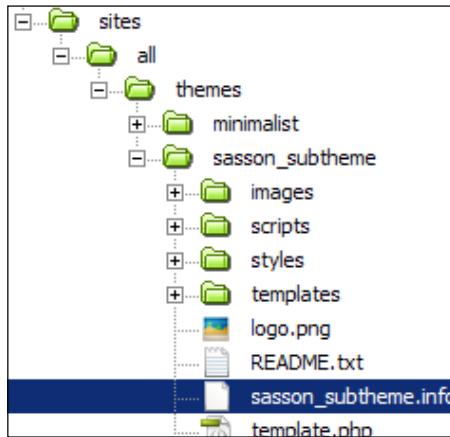
We begin by installing the Sasson theme. We then create the subtheme from subtheme prototype, and finally we will configure the subtheme and configure it to display only for mobile devices.

1. Go to the Sasson project page: <http://drupal.org/project/sasson>.
2. Copy the tar.gz URL of the latest **Recommended release** of the Sasson theme:

Recommended releases				
Version	Downloads	Date	Links	
7.x-2.0	tar.gz (317.81 KB) zip (466.08 KB)	2012-Feb-19	Notes	
Development releases				
Version	Downloads	Date	Links	
7.x-2.x-dev	tar.gz (317.81 KB) zip (466.09 KB)	2012-Feb-20	Notes	

3. Go to your Drupal site and select **Module** from the admin menu.
4. Select **+Install new module**.
5. Paste the URL in the **Install from URL** field, then click on **Install**.
6. In your FTP client, go to `Drupal root/sites/all/themes/sasson`.
7. Copy the folder `SUBTHEME` to the folder `Drupal root/sites/all/themes/`.
8. Rename the `SUBTHEME` folder to `sasson_subtheme`, and the copied theme folder should now have the path `Drupal root/sites/all/themes/sasson_subtheme`.

9. In the sasson_subtheme folder, rename the file SUBTHEME.info to sasson_subtheme.info:



10. Open the file sasson_subtheme.info.
 11. Edit the **name** field to read as follows:
`name = Sasson subtheme`
 12. Save and upload the file.
 13. Select **Appearances** from the admin menu.
 14. Scroll to the **Disabled themes** section and select **Enable** for the **Sasson** theme and also for the **Sasson subtheme**.
 15. Select **Configuration** from the admin menu, then select **Mobile tools**.
 16. Select the **Theme switching** tab.
 17. Ensure that the option **Switch theme for a mobile device** is selected.
 18. Select the theme **sasson_subtheme** from the **Mobile theme** field:

When do you want to switch themes
<input type="radio"/> No theme switch <input checked="" type="radio"/> Switch theme for a mobile device * <input type="radio"/> Switch theme based on the URL
Mobile theme
sasson_subtheme ▾ Select your default mobile theme. You can specify a different theme for different devices.

19. Click on **Save configuration**.
20. Load your site on a mobile device to test.

How it works...

The first step we take is to install the Sasson theme using the regular module installer. We then go to the theme's folder and copy the **SUBTHEME** folder into the theme's root folder. The **SUBTHEME** folder is the prototype for a Sasson subtheme. It should not be edited directly. We rename the copied **SUBTHEME** folder to **sasson_subtheme**.

After copying and renaming the **SUBTHEME** folder, we rename the theme's info file so that its name matches the folder's name. We then open the info file and rename the name attribute to Sasson subtheme. This is the human readable name for the theme.

After creating the subtheme, we go to the **Appearances** page and enable both the Sasson base theme and the Sasson subtheme. Once both themes are enabled, they are ready to use, so we just need to configure the site to load the Sasson subtheme when the site is being rendered to a mobile device. To do this, we go to the **Mobile tools** module and set the **sasson_subtheme** as the default theme under the Mobile theme setting.

Finally, we test the theme in a mobile browser. The width of the content will resize to fit the dimensions of the screen and adapt when the orientation is changed between landscape and portrait. The theme is now ready to start theming.

There's more...

The Sasson theme has many great features that we haven't covered in this recipe, such as SASS and Google fonts support. There are also a variety of other themes that can provide very good mobile support.

Theme display breakpoints

The Sasson theme provides you with the option to set custom layout breakpoints. This is to say that you can enter screen pixel widths at which the layout of the page will switch to a different layout. For example, for all screen widths up to 480px, there will be no left or right column.

Support for SASS

The Sasson theme contains in-built support for the CSS3 extension SASS, which stands for Syntactically Awesome Stylesheets. SASS allows you to write and manage stylesheets with staggering efficiency because it allows you to use variables in your CSS code. For example, you can define a color, and use the variable to reference the color later on:

```
$companyBlue = # 006688;  
.header {background:$companyBlue;}
```

SASS also allows you to define CSS blocks that inherit definitions from other blocks, and nesting of selectors, which greatly reduces the repetition of code.

Support for Google fonts

The Sasson theme contains support for the Google web fonts API. Google web fonts expand the range of fonts available to the browser for use in CSS styling. To use a Google web font with the Sasson theme, simply include the name of the required font in the font section of the theme's configuration page.

Other adaptive themes

There are numerous alternative themes available that provide responsive mobile and tablet layouts. The theme you use will depend on your requirements; you might want to evaluate the following alternatives:

- ▶ AdaptiveTheme, at <http://drupal.org/project/adaptivetheme>
- ▶ Fusion, at <http://drupal.org/project/fusion>
- ▶ Omega, at <http://drupal.org/project/omega>

See also

- ▶ Sasson, smart Drupal theming, at <http://www.linnovate.net/blog/sasson-smart-drupal-theming>
- ▶ Sass, at <http://sass-lang.com/>
- ▶ Google web fonts, at <http://www.google.com/webfonts>

Configuring theme compression and caching

Webservers can quickly become overloaded by multiple users requesting pages that require multiple queries. Furthermore, pages that consist of lots of CSS and JavaScript files can reduce the page load time. Both of these issues can be frustrating to the end-user.

In this recipe, we will see how to enable and configure Drupal's caching, which can dramatically reduce the server response time. We will also learn how to activate aggregation of the theme's CSS files, a process that can greatly reduce the number of requests to the server per page load, and vastly improve the page load time for the end-user.

How to do it...

To begin, we will enable and configure Drupal's caching capabilities. We will then enable and configure the optimization options for the theme's CSS and JavaScript files.

1. Select **Configure** from the admin menu, then select **Performance**.
2. In the **CACHING** section check the option **Cache pages for anonymous users**.
3. Check the option **Cache blocks**.
4. In the **Minimum cache lifetime** field, select **5 min**.
5. In the **Expiration of cached pages** field, select **1 hour**.
6. Click on **Save configuration**.
7. In the **BANDWIDTH OPTIMIZATION** section, check the option **Compress cached pages**.
8. Check the option **Aggregate and compress CSS files**.
9. Check the option **Aggregate JavaScript files**.

The screenshot shows the 'Performance' configuration page in Drupal. It has three main sections: 'CLEAR CACHE' (with a 'Clear all caches' button), 'CACHING' (with checkboxes for 'Cache pages for anonymous users' and 'Cache blocks', and dropdowns for 'Minimum cache lifetime' set to '5 min' and 'Expiration of cached pages' set to '<none>'), and 'BANDWIDTH OPTIMIZATION' (with checkboxes for 'Compress cached pages', 'Aggregate and compress CSS files.', and 'Aggregate JavaScript files.' all checked). The 'CACHING' section also includes a note: 'Cached pages will not be re-created until at least this much time has elapsed.'

10. Click on **Save configuration**.

How it works...

The first option we select is **Cache pages for anonymous users**. This option will store the generated HTML of a page in the database, reducing the number of database queries required to serve the page, and therefore reducing the server load. This option only applies to users who are not logged in. For logged in users, pages are re-created on each page load.

We check the option **Cache blocks**. This option causes cacheable blocks to be cached for each combination of user roles. Enabling this option will enable block caching for all users, provided that the block has been designed to allow caching.

We set the **Minimum cache lifetime** to **5 minutes**. This means that cached content is kept for at least 5 minutes, unless the cache is cleared manually. The time chosen for the minimum cache lifetime should be dependent on your website's needs. A low time of 1 minute is preferable for websites with lots of news content, so the cache will be refreshed after one minute. However, this can place quite a strain on a busy server, and you may need to experiment with the values to find a value that works for you.

The option, **Expiration of cached pages** relates to external caching by browsers and other systems such as firewalls. We set this option to **1 hour**; however, you may find that on your website you may need to reduce this value if you are having a problem with content being unnecessarily cached.

We then select **Save configuration** before setting the **BANDWIDTH OPTIMIZATION** settings; this enables all three of the optimization features.

Drupal contains three very useful optimization options, **Compress cached pages**, **Aggregate and compress CSS files**, and **Aggregate JavaScript files**. We enable all of these options. When enabled, these options compress the page transmitted to the browser, and aggregate the potentially large number of CSS files and JavaScript files into a lower number of larger files. This is good because it reduces the number of requests to the server, and therefore decreases page-load time.



It's generally recommended to only switch on caching and aggregation when your site is live so that any changes made in theme development or site building are noticeable immediately.

There's more...

In the above configuration we have seen how we can provide a significant performance boost for most websites. If you are rolling out a very large-scale site, then you will need to look at something even more powerful.

The Varnish HTTP accelerator

Varnish Cache is an application that can provide performance boosts of between 300-1000 times the previous speed. Varnish is an application that is installed on the webserver, and acts as an intermediary between the user's request and the Drupal application.

Furthermore, there is a Drupal module that provides an interface to the Varnish application's administrative functions:

- ▶ <https://www.varnish-cache.org/>
- ▶ <http://drupal.org/project/varnish>

The Boost module

The Boost module provides improved caching performance for serving static pages to anonymous users. Boost is easy to install, and it allows the possibility of having different cache lifetimes for particular pages. For more information refer to the following site:

- ▶ <http://drupal.org/project/boost>

10

Working with Other Languages

In this chapter we will cover:

- ▶ Installing another language using Locale
- ▶ Managing interface translation using Locale
- ▶ Enabling content type translation
- ▶ Displaying a language switching block for end users
- ▶ Creating a multilingual View

Introduction

Drupal provides an intuitive method of managing multilingual content as part of its core functionality. In this chapter, we will see how to configure the Locale module so that the site can function in multiple languages. The Locale module, in conjunction with an enabled language file, can translate the interface into the user's current language. We will see how the interface translations can be tweaked, or added if they are missing.

We then move on to content translation. Each content type has the option to be translatable; we will configure the Basic page content type so that it can be translated, and provide a sample translation.

When providing multilingual content, it's important to provide a mechanism for the users to select their language of choice, so we will see how to configure the language-switching block for end users.

To finish the chapter, we will look at the more advanced example of providing translations for a view. We will see how to create a view out of a translatable content type, and configure it so that it automatically displays the list items in the user's language of choice.

Installing another language using Locale

In this recipe, we will be enabling and configuring the **Locale** module that provides the multi-language translations functionality in Drupal. It also provides us with the ability to manage the translations of the interface. We will install the French language, and then configure the site so that it switches language when the language string in the URL changes.

Getting ready

To complete this recipe, please enable the Locale module in the Drupal core. We will also be making use of the Localization update module. Please install and enable the module from this location:

- ▶ http://drupal.org/project/l10n_update

How to do it...

To begin, we will add a new language to the available language list and then configure the language detection options for the site. Finally, we will update the language from the language server.

1. Select **Configuration** from the admin menu.
2. Find the **REGIONAL AND LANGUAGE** section, then select **Languages**.
3. Select **+Add language**.
4. Choose the option **French (Français)** from the **Language name** field:

Add a language to be supported by your site. If your desired language is not available in the *Language name* drop-down, click *Custom language* and provide a language code and other details manually. When providing a language code manually, be sure to enter a standardized language code, since this code may be used by browsers to determine an appropriate display language.

▼ PREDEFINED LANGUAGE

Language name

Use the *Custom language* section below if your desired language does not appear in this list.

► CUSTOM LANGUAGE

5. Click on **Add language:**

With multiple languages enabled, interface text can be translated, registered users may select their preferred language, and authors can assign a specific language to content. [Download contributed translations](#) from Drupal.org.

[+ Add language](#)

ENGLISH NAME	NATIVE NAME	CODE	DIRECTION	ENABLED	DEFAULT	OPERATIONS
⊕ English	English	en	Left to right	<input checked="" type="checkbox"/>	<input checked="" type="radio"/>	edit
⊕ French	Français	fr	Left to right	<input checked="" type="checkbox"/>	<input type="radio"/>	edit delete

[Save configuration](#)

6. Select the **DETECTION AND SELECTION** tab.

7. Check the **ENABLED** option for the **URL** row, then click on **Save settings:**

User interface text language detection

Order of language detection methods for user interface text. If a translation of user interface text is available in the detected language, it will be displayed.

DETECTION METHOD	DESCRIPTION	ENABLED	OPERATIONS
⊕ URL	Determine the language from the URL (Path prefix or domain).	<input checked="" type="checkbox"/>	Configure
⊕ Session	Determine the language from a request/session parameter.	<input type="checkbox"/>	Configure
⊕ User	Follow the user's language preference.	<input type="checkbox"/>	
⊕ Browser	Determine the language from the browser's language settings.	<input type="checkbox"/>	
⊕ Default	Use the default site language (English).	<input checked="" type="checkbox"/>	

8. Select the **TRANSLATE UPDATES** tab then select the link **Translation update administration page**.

Working with Other Languages

9. Ensure that the option **All existing translations are kept, only new translations are added** is selected:

List of latest imported translations and available updates for each enabled project and language.

If there are available updates you can click on Update for them to be downloaded and imported now or you can edit the configuration for them to be updated automatically on the [Update settings page](#)

Drupal core

drupal 7.12	Update available
-------------	------------------

Modules

Localization update 7.x-1.0-beta3	Update available
LANGUAGES	

Update mode

Translation updates replace existing ones, new ones are added
 Edited translations are kept, only previously imported ones are overwritten and new translations are added
 All existing translations are kept, only new translations are added.

[Update translations](#) [Refresh information](#)

10. Click on **Update translations**.

How it works...

We begin by adding a new language that can be used by the system. We do this simply by selecting a language from a list of available languages.

When adding a new language, the language is configured with some default values. In the case of the French language, the following default values are used:

Language name in English	French
Native language name	Français
Path prefix language code	fr
Direction	Left to right

After adding a new language, we then configure the **Detection and Selection** options.

By enabling the **URL** detection method, we delegate the system's primary language detection mechanism to the URL. For example, if we were to have the URL `http://example.com/node/1`, then the French translation of this page would be at `http://example.com/fr/node/1`.

Next, we see how to use the translation update service. By selecting to **Update translations**, we cause the site to download the latest translations for the installed languages, if any exist.

Finally, to test the new language installation, go to the root of your site, with the **fr** suffix in the URL, for example, `Drupal root/fr/`. You will now see the homepage which has its interface text translated. However, the content is not translated. This has to be done manually, and we will see how to do this later in the chapter.

There's more...

So far, we have seen how to implement a typical installation. But what if you want a language that isn't on the list? Or to use a different language selection method?

Adding a custom language

When adding a new language, there is an option to add a **Custom language**. This option allows you to add the language code, language prefix, and path. Once the custom language is added, Drupal allows you to translate the interface elements as you see fit.

Other language detection methods

The example in this recipe uses the **URL** method of language detection, which detects the language based on the language path entered in the URL.

There are other methods of detection such as **Session**, which determines the current language from a session variable that stores the user's most-recent language selection preference.

There is also the option to use the **User** detection method, which uses the user's preferred language, if one is set. And finally, there is an option to determine the language by obtaining the web Browser's language.

It's possible to use any combination of these options, and you are able to set an order of preference in which they are invoked.

Managing interface translation using Locale

Drupal provides excellent options for translating its interface, with a wide range of languages already implemented; however, the translations are often not complete. Using the translation interface, it's possible to add the missing translations, and also to modify any translations that you might find unsuitable or incorrect. In this recipe, we will take a look at the translation interface provided by Drupal's Locale module and apply a change to an existing translation from the French language file.

Getting ready

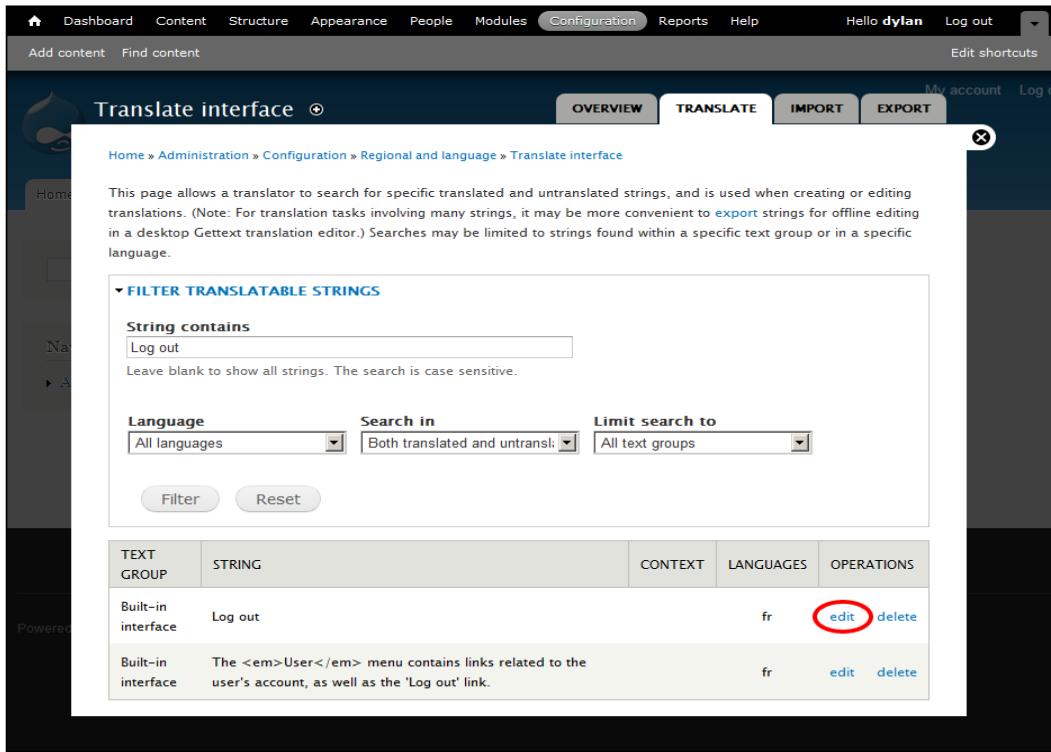
To complete this module, you will need to have enabled the Locale module in the Drupal core, and installed the French language file. See the preceding recipe for instructions on how to do this.

How to do it...

In this recipe, we will be translating the French version of the string "Log out", which exists on the administration menu bar:

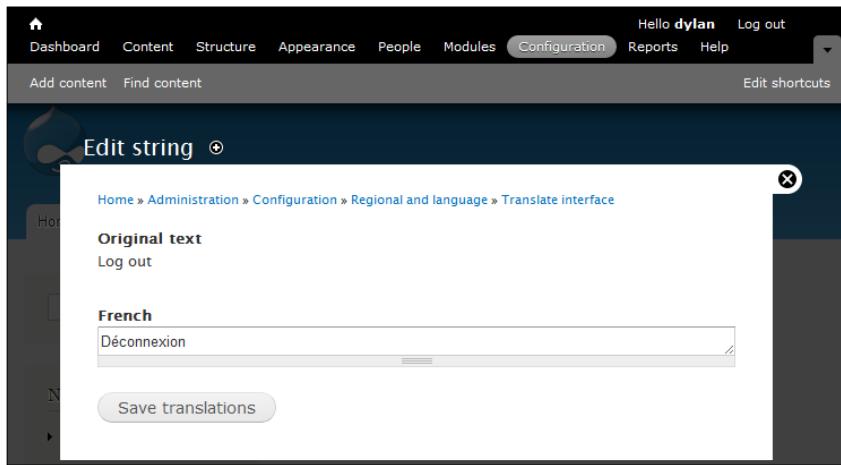
1. Select **Configuration** from the admin menu, then select **Translate interface** in the **Regional and Language** section.
2. Select the **TRANSLATE** tab.
3. In the field **String contains**, enter the search term **Log out** and click on **Filter**.

4. Select **edit** in the table for the **Log out** string:



The screenshot shows the 'Translate interface' page. At the top, there are tabs for 'OVERVIEW', 'TRANSLATE' (which is selected), 'IMPORT', and 'EXPORT'. Below the tabs, there's a search bar with 'String contains' set to 'Log out'. Under 'Language', 'Search in' is set to 'Both translated and untranslated', and 'Limit search to' is set to 'All text groups'. A 'Filter' and 'Reset' button are below these settings. The main area is a table titled 'Edit string' with columns: TEXT GROUP, STRING, CONTEXT, LANGUAGES, and OPERATIONS. There are two rows in the table. The first row has 'Built-in interface' in the TEXT GROUP column, 'Log out' in the STRING column, 'fr' in the LANGUAGES column, and 'edit' and 'delete' in the OPERATIONS column. The second row has 'Built-in interface' in the TEXT GROUP column, a longer string in the STRING column, 'fr' in the LANGUAGES column, and 'edit' and 'delete' in the OPERATIONS column. The 'edit' link in the first row is circled in red.

5. Change the **Translation** field titled **French** to read: **Déconnexion**:



The screenshot shows the 'Edit string' page. At the top, there are tabs for 'OVERVIEW', 'TRANSLATE' (selected), 'IMPORT', and 'EXPORT'. Below the tabs, there's a section for 'Original text' with 'Log out' listed. Under 'French', the translation field contains 'Déconnexion'. A 'Save translations' button is at the bottom.

6. Click on **Save translations**.

How it works...

We open the **Translate** tab in the **Translate interface** popup, which is the dashboard for all of the strings of text on the site that have been made available for translation. We then find the string "Log out" and edit it by changing the French version of the string to the shorter word **Déconnexion**. We pick this example merely as a demonstration of how the translation of strings is accomplished, but translating this particular string can help to reduce the width of the buttons on the admin bar, which in some cases can cause the buttons to go onto two lines.



In Drupal, strings of text can be made available to the translation engine by wrapping the string inside the t () function.



See also

- ▶ Translating a site interface into different languages, at
<http://drupal.org/documentation/modules/locale>

Enabling content type translation

In this recipe, we will configure the Basic page content type so that it is translatable, and then add some content in both English and French.

Getting ready

To complete this module, you will need to have enabled the Locale module in the Drupal core, and installed the French language file. See the first recipe of this chapter for instructions on how to do this (*Installing another language using Locale*).

You will also need to enable the Drupal native module, **Content translate**.

How to do it...

We will start by making the Basic page content type translatable; we will then go and create a new Basic page content node in English. We will then add a French translation to it.

1. Select **Structure** from the admin menu, then select **Content types**.
2. Select **edit** for the **Basic page** content type.
3. Select the **Publishing options** tab.

4. In the **Multilingual support** field, select **Enabled, with translation**:

Multilingual support

Disabled

Enabled

Enabled, with translation ←

Enable multilingual support for this content type. If enabled, a language selection field will be added to the editing form, allowing you to select from one of the **enabled languages**. You can also turn on translation for this content type, which lets you have content translated to any of the installed languages. If disabled, new posts are saved with the default language. Existing content will not be affected by changing this option.

5. Click on **Save content type**.
6. Select **Add content** from the admin shortcuts menu.
7. Select **Basic page** from the list of available content types.
8. In the **Title** field, enter **Welcome**.
9. In the **Body** field, enter: **Welcome to the English language version of our page**.
10. In the **Language** field, select **English**:

Title *
Welcome

Body (Edit summary)
Welcome to the English language version of our page.

Text format Filtered HTML ▾ More information about text formats ?

- Web page addresses and e-mail addresses turn into links automatically.
- Allowed HTML tags: <a> <cite> <blockquote> <code> <dl> <dt> <dd>
- Lines and paragraphs break automatically.

Language
English ▾

11. Click on **Save**.

Working with Other Languages

12. Select the **Translate** tab for the node:

The screenshot shows a node titled "Welcome". Below the title are three buttons: "View", "Edit", and "Translate". The "Translate" button is circled in red. The main content area contains the text: "Welcome to the English language version of our page."

13. Select **add translation** for the **French** row:

The screenshot shows a list of translations for the "Welcome" node. The columns are: LANGUAGE, TITLE, STATUS, and OPERATIONS. There are two rows: one for "English (source)" with status "Published" and operation "edit"; and one for "French" with status "Not translated" and operation "add translation", which is circled in red.

LANGUAGE	TITLE	STATUS	OPERATIONS
English (source)	Welcome	Published	edit
French	n/a	Not translated	add translation

14. In the **Title** field, enter **Bienvenue**.

15. In the **Body** field, enter: **Bienvenue sur la version française de notre page**:

The screenshot shows the edit form for the "Bienvenue" translation. It includes fields for "Title" (containing "Bienvenue"), "Body" (containing "Bienvenue sur la version française de notre page"), "Format de texte" (set to "Filtered HTML"), and "Langue" (set to "Français"). A note below the body field specifies supported HTML tags and line/paragraph handling.

Title *
Bienvenue

Body (Modifier le résumé)
Bienvenue sur la version française de notre page

Format de texte Plus d'information sur les formats de texte

• Les adresses de pages web et de courriels sont transformées en liens automatiquement.
• Tags HTML autorisés : <a> <cite> <blockquote> <code> <dl> <dt> <dd>
• Les lignes et les paragraphes vont à la ligne automatiquement.

Langue
Français

16. Select **Enregister** to save the translation.

How it works...

To begin, we configure the Basic page content type and switch on content translation by setting its **Multilingual support** field to **Enabled, with translation**. This will enable the language selection field for the content. It will also enable the translation interface.

Next we create a new item of Basic page content, and save it. We then open the **Translation** tab for the node, and enter the French equivalent of the content.

After saving the French version of the article, the translated content is now ready. We can test this out by changing the URL language suffix; for example, Drupal root/node/1 will display the English version of the node, and Drupal root/fr/node/1 will display the French version.

[ The actual node ID of the Basic page you have created will probably be different, depending on how many items of content you've already added.]

This solution is obviously not a convenient means for the user to set the language; we will see how to provide translation buttons for the user in the next recipe.

See also

The next recipe, *Displaying a language switching block for end users*.

Content Translation core module, at <http://drupal.org/documentation/modules/translation>.

Displaying a language switching block for end users

In the previous recipes of this chapter we have seen how to configure the site and its content types for translation. However, we have not provided the users with a mechanism for switching language. In this recipe, we will enable the language-switching block so that end users can easily switch to their language of choice.

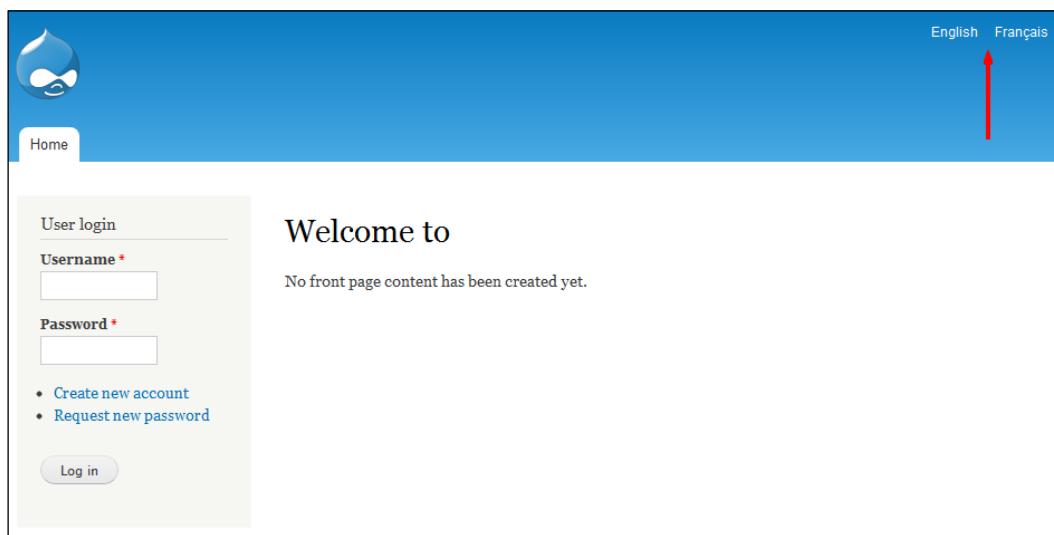
Getting ready

This recipe requires that you first complete the previous recipe, adding translation functionality to a content type.

How to do it...

In this recipe we are simply going to enable and select a region for the language switching block:

1. Select **Structure** from the admin menu, then select **Blocks**.
2. Find the **Language switcher** block and set its **REGION** to **Header**.
3. Click on **Save blocks**.
4. Select the home button to view the language switcher.



How it works...

The language-switching block is part of the Locale module in the Drupal core, so all we need to do is set it to a region. We do this by opening the Blocks listing page and setting the region to Header. We choose the **Header** region because this is a place where language translation buttons are commonly found on other sites.

There's more...

For many use cases, the language switching block seen in this recipe is more than adequate, but there are some UI improvement modules that are compatible with the native language functionality.

Language icons

This is a module that adds language icons to the language switcher block. It will also add icons to the language switcher links that are output by translated nodes.

- ▶ <http://drupal.org/project/languageicons>

Language switcher drop-down

This module creates a new block on the **Blocks** page, which provides the language options in a drop-down, rather than in a list of links. When used in conjunction with the Language icons module, this module will output the language options next to their icons.

- ▶ http://drupal.org/project/lang_dropdown

Creating a multilingual View

In this recipe we will be creating a multilingual view by building upon what we have already learnt in this chapter, and making use of the Internationalization modules, which give us the ability to create translatable views.

Getting ready

This recipe builds upon all of the previous recipes in the chapter, and it also requires that you install the latest releases of the following modules:

- ▶ <http://drupal.org/project/i18n>
- ▶ <http://drupal.org/project/i18nviews>
- ▶ <http://drupal.org/project/variable>
- ▶ <http://drupal.org/project/views>
- ▶ <http://drupal.org/project/ctools>

Once installed, enable the following features:

- ▶ Chaos tools
- ▶ Internationalization
- ▶ Internationalization Views
- ▶ String translation
- ▶ Variable
- ▶ Views
- ▶ Views UI

How to do it...

We will begin by creating a new basic view, which just outputs a list of all content on the site, in teaser form. Then, we will add a filter to the view which tells the view to bring back content in the user's chosen language. Finally, we will save and test the new view.

1. Select **Structure** from the admin menu, then select **Views**.
2. Select **+Add new view**.
3. In the **View name** field, enter **Content list**.
4. Leave the default settings in the firstfieldset.
5. Also leave the secondfieldset configured in its default state.

The screenshot shows the 'Add new view' configuration page. The 'View name' is set to 'Content list'. Under 'Description', the 'Show' dropdown is set to 'Content' and 'of type' is 'All'. The 'sorted by' dropdown is set to 'Newest first'. Under 'Create a page', the 'Page title' is 'Content list' and the 'Path' is 'http://drupal7cbc10.streetfish.co.uk/content-list'. The 'Display format' section has 'Unformatted list' selected, along with 'teasers', 'with links (allow users to add comments, etc.)', and 'without comments'. The 'Items to display' is set to 10, and 'Use a pager' is checked. Under 'Create a block', there are two empty checkboxes.

6. Click on **Continue & edit**.
7. In the **FILTER CRITERIA** section, click on the **add** button.
8. In the **Search** field, enter the search term, **language**, to filter the range of options.

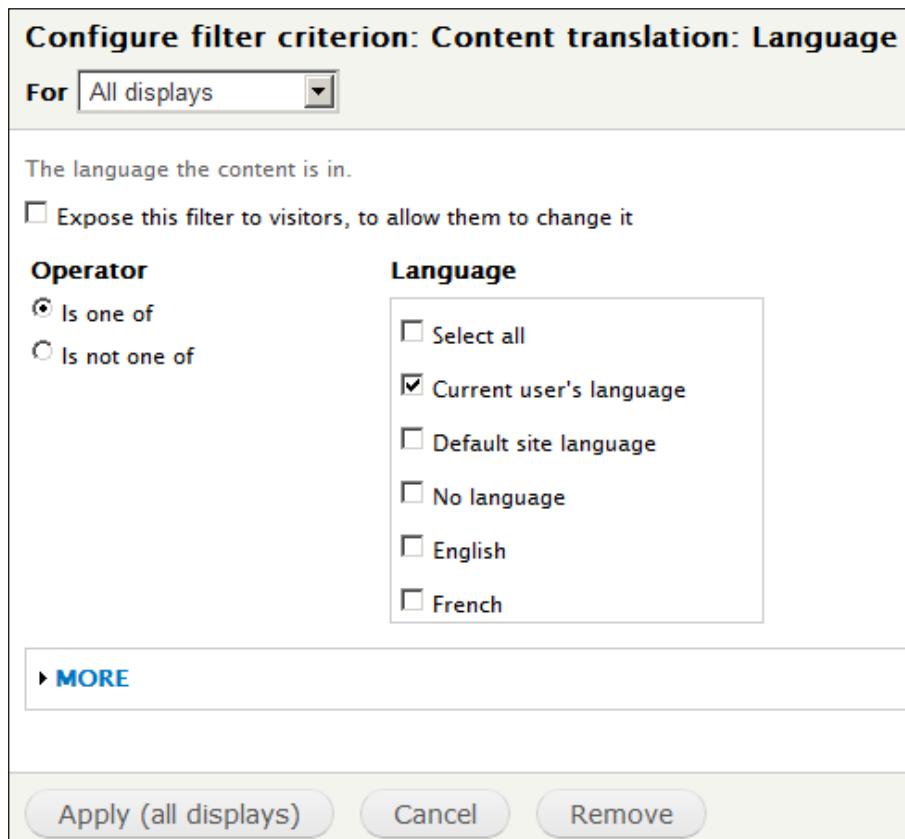
9. Check the option **Content translation: Language**:



A screenshot of a search interface. At the top, there is a search bar labeled "Search" containing the word "language" and a filter dropdown labeled "Filter" set to "- All -". Below the search bar, there is a list item with a checked checkbox next to the text "Content translation: Language". A tooltip below the checkbox says "The language the content is in."

10. Select **Apply (all displays)**.

11. In the following pop up, ensure the **Operator** field is set to **Is one of**.
12. In the **Language** field, check the option **Current user's language**:



The dialog box title is "Configure filter criterion: Content translation: Language". The "For" dropdown is set to "All displays".
Operator: Is one of (radio button selected).
Language:
Select all
Current user's language (checkbox checked)
Default site language
No language
English
French
Buttons: MORE, Apply (all displays), Cancel, Remove.

13. Select **Apply (all displays)**.

Working with Other Languages

The screenshot shows the 'Page details' configuration page for a view named 'Page'. The page is divided into several sections:

- TITLE:** Title: Content list
- FORMAT:** Format: Unformatted list | Settings
Show: Content | Teaser
- FIELDS:** The selected style or row format does not utilize fields.
- FILTER CRITERIA:** Content: Published (Yes)
Content translation: Language (= Current user's language)
- SORT CRITERIA:** Content: Post date (desc)
- PAGE SETTINGS:** Path: /content-list
Menu: No menu
Access: Permission | View published content
- HEADER:** add
- FOOTER:** add
- PAGER:** Use pager: Full | Paged, 10 items
More link: No

14. Click on **Save**.

15. The View is now complete, and you can test this by going to Drupal root/ content-list and then by changing the language to **French**, with the language selection block.

How it works...

First, we add a simple view that we create with all of the default options. This will output a list of **All** content types, sorted by **Newest first**. The view will also display the result as a list of **Unformatted teasers, with links**.

After continuing to the view settings page, we add a **Language** filter, and configure it so that the language is the same as the user's language.

There's more...

In this recipe, we barely touch the surface of what's possible with the Internationalization set of modules. There are a wide range of features that are out of the scope of this book.

Other features of the Internationalization set of modules

The Internationalization module enables extra functionality that isn't included in Drupal's core modules such as translation abilities for menus, taxonomies, and URL paths.

The Internationalization module provides the functionality to translate blocks, and to configure their visibility according to the language. The module also provides features such as Forum translation and Contact translation.

See also

- ▶ Internationalization (i18n) module, at <http://drupal.org/node/133977>

11

Managing Users

In this chapter we will cover:

- ▶ Creating new user accounts
- ▶ Managing user roles
- ▶ Setting up a new user notification
- ▶ Adding a biography field to the user profile
- ▶ Building a grid view of profile pictures

Introduction

Administrating any CMS-based website inevitably requires a degree of involvement with user management. In this chapter, we will cover the various topics related to managing users, starting with a recipe on *Creating new user accounts*, and then *Managing user roles*, where we see how to create new user roles with bespoke permissions, and assign users to that role.

In the recipe *Setting up a new user notification*, we will see how to use Drupal's Trigger and Action functionality to send a notification to administrator users when a new user registers.

We then look at expanding the fields that are available in the user profile in the recipe *Adding a biography field to the user profile*. This recipe demonstrates the flexible way that Drupal user profiles can be expanded and customized for almost any use.

We finish with a recipe, *Building a grid view of profile pictures*, which further expands the user profile and makes use of the Views module to create a staff listing display that outputs the user name, profile picture, and job title of employee users, in a 4x4 grid.

Creating new user accounts

Invariably you will at some point need to create new user accounts. Of course, it is possible to allow the user to simply register for their new account themselves, but this is not always appropriate. In this recipe, we will see how an admin user can create an active user account for another user.

Getting ready

To complete this recipe you will need an e-mail address that hasn't been used by any other users in the system.

How to do it...

In this simple recipe we will simply be creating a new dummy user account:

1. Select **People** from the admin menu, then select **+Add user**.
2. In the **Username** field, enter **dummyuser1**.
3. In the **E-mail address** field, enter your new e-mail address for the user.
4. In the **Password** field, enter a suitable password, at least six characters long.
5. In the **Confirm password** field, re-enter the password entered in the previous step.
6. Leave the **Status** field set as **Active**.
7. Do not select any extra **Roles**, leaving just **authenticated user** checked.

8. Check the option, **Notify user of new account**:

Username *

Spaces are allowed; punctuation is not allowed except for periods, hyphens, apostrophes, and underscores.

E-mail address *

A valid e-mail address. All e-mails from the system will be sent to this address. The e-mail address is not made public and will only be used if you wish to receive a new password or wish to receive certain news or notifications by e-mail.

Password *
 Password strength: Fair

Confirm password *
 Passwords match: yes

To make your password stronger:
• Add uppercase letters
• Add numbers
• Add punctuation

Provide a password for the new account in both fields.

Status
 Blocked
 Active

Roles
 authenticated user
 administrator
 Notify user of new account

Create new account

9. Click on **Create new account**.

How it works...

To start, we go to the **People** menu and add a new user by creating a new username and password. We ensure that the **Status** is set to **Active**; this will mean that the account is immediately ready to use.



When a user self-registers through the registration form, the system is often set up to require admin approval, or simply e-mail address approval. In those cases, the account would remain in the **Blocked** state until approval.

We don't select any **Roles**, as there is only one additional role, apart from Administrator, that we can select. Later on in the chapter we will see how to create and apply user roles.

Finally, we select **Notify user of new account**. This causes an e-mail to be sent to the new user account's e-mail address, notifying them of their new account.

There's more...

This recipe shows the very simple procedure of creating a user account, but there are other ways to do this that may be useful.

Automatically creating user accounts

The Devel module provides a feature called **Generate users**, which you can use to generate any number of user accounts with a user role that you specify. The generator will create the specified number of accounts, each with a randomly generated username.

Creating user accounts from CSV

Using the Feeds module, it's possible to import a list of user accounts from a CSV spreadsheet file. This method can be a very useful tool when migrating from one system to another, or if a list of users needs to be managed outside of the Drupal site.

Managing user roles

User roles are very useful for providing varied degrees of access to particular features or areas of your site. Drupal makes it very easy to set up roles for editing, creating, or just viewing content, and in this recipe we will be setting up a new **Editor** role which we will configure to have the permission only to edit the Basic page and Article content.

Getting ready

As part of this recipe, we will be creating a new user account. You will need an e-mail address that has not already been used by a user account in the system.

How to do it...

We will begin by creating the new role; we will then apply the permissions for the new role. Finally, we will create an editor user and assign the Editor role to the user.

1. Select **People** from the admin menu, then select the **Permissions** tab.
2. Select the smaller **Roles** tab below the **Permissions** tab.
3. At the bottom of the roles table, enter **Editor** in the new roles field:

NAME	OPERATIONS	
⊕ anonymous user (<i>locked</i>)		edit permissions
⊕ authenticated user (<i>locked</i>)		edit permissions
⊕ administrator	edit role	edit permissions
Editor	Add role	

4. Click on **Add role**:

NAME	OPERATIONS	
⊕ anonymous user (<i>locked</i>)		edit permissions
⊕ authenticated user (<i>locked</i>)		edit permissions
⊕ administrator	edit role	edit permissions
⊕ Editor	edit role	edit permissions
	Add role	

5. Select **edit permissions** for the **Editor** role.
6. Find and check the following permissions:
 - Access the content overview page**
 - Article: Edit any content**
 - Basic page: Edit any content**
 - View the administration theme**
 - Use the administration toolbar**

Managing Users

7. Click on **Save permissions**.
8. Select **People** from the admin menu, then select **+Add user**.
9. In the **Username** field, enter **editor1**.
10. In the **E-mail address** field, enter your e-mail address for the new account.
11. In the **Password** field, enter a password for the new account; ensure that it's at least six characters long.
12. Re-enter your password in the **Confirm password** field.
13. Ensure that **Active** is selected for the **Status** option.
14. In the **Roles** section, check the **Editor** role.
15. Check the option, **Notify user of new account**:

The screenshot shows the 'Add User' form. The 'Username' field contains 'editor1'. Below it is a note: 'Spaces are allowed; punctuation is not allowed except for periods, hyphens, apostrophes, and underscores.' The 'E-mail address' field contains 'editor1@example.com'. A note below it says: 'A valid e-mail address. All e-mails from the system will be sent to this address. The e-mail address is not made public and will only be used if you wish to receive a new password or wish to receive certain news or notifications by e-mail.' The 'Password' field is empty, and the 'Password strength' indicator shows 'Weak'. The 'Confirm password' field is also empty. Below these fields is a note: 'To make your password stronger:' followed by a bulleted list: '• Make it at least 6 characters', '• Add lowercase letters', '• Add uppercase letters', '• Add numbers', and '• Add punctuation'. A note below the password fields says: 'Provide a password for the new account in both fields.' Under the 'Status' section, the 'Active' radio button is selected. In the 'Roles' section, several checkboxes are checked: 'authenticated user', 'Administrator', 'Editor', and 'Notify user of new account'.

Username *
editor1
Spaces are allowed; punctuation is not allowed except for periods, hyphens, apostrophes, and underscores.

E-mail address *
editor1@example.com
A valid e-mail address. All e-mails from the system will be sent to this address. The e-mail address is not made public and will only be used if you wish to receive a new password or wish to receive certain news or notifications by e-mail.

Password *
 Password strength: **Weak**

Confirm password *

To make your password stronger:

- Make it at least 6 characters
- Add lowercase letters
- Add uppercase letters
- Add numbers
- Add punctuation

Provide a password for the new account in both fields.

Status

Blocked

Active

Roles

authenticated user

administrator

Editor

Notify user of new account

16. Click on **Create new account**.

How it works...

We begin by creating a new user role, which we call Editor. This role is intended to be used only for editing content, and not for creating or deleting it. It will not be given any of the other permissions, and will therefore be quite limited.

After creating the role, we edit its permissions. We enable **Access the content overview page**. This enables the role to view the list of content items available for editing. We also enable **Article: Edit any content**, and **Basic page: Edit any content**. These permissions enable the Editor role just to edit all content of Article and Basic page types.

We then enable the permission **View the administration theme**. This permission is not essential, but it can help in providing a consistent interface for carrying out admin tasks. We also enable **Use the administration toolbar**. This gives the Editor users access to the admin toolbar with a shortcut to the **Content overview** page.

After configuring the permissions for the Editor role, we create a new user **editor1**, who we assign the Editor role. After saving the new user, log out of the site and log in using the editor1 account. You will now see you are only able to view the **Content overview** page and edit content of the Article and Basic page types.

Setting up a new user notification

In Drupal, there are a number of automatically generated e-mails which are sent to users at specific trigger points; for example, when a user registers, is approved, or is blocked. However, if you are an admin user and want to receive a notification when a user has registered on your site in order to go and approve their account, you need to set this up manually.

In this recipe, we will be using Drupal's native Trigger module to set up an e-mail notification to an admin user when a new user registers.

Getting ready

To complete this recipe you will need to enable the **Trigger** module, which is available as part of the Drupal core.

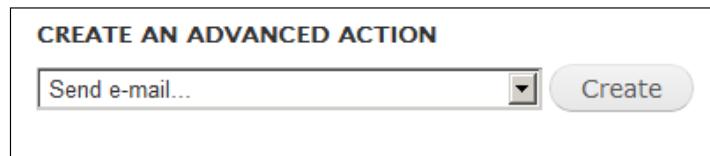
You will also need to install and enable the Token module:

- ▶ <http://drupal.org/project/token>

How to do it...

We will start by creating a new action that will be used to send the e-mail to the admin user; we will then configure this action to be activated on the new user trigger:

1. Select **Configuration** from the admin menu, then select **Actions**.
2. In the section **CREATE AN ADVANCED ACTION**, choose the option **Send e-mail...**:



3. Click on **Create**.
4. In the **Label** field of the **Configure an advanced action** popup, update the **Label** field to read **Send new user notification email to admin**.
5. Select the **Recipient** field and scroll down to the tokens table.
6. Expand the **Site information** row and select the option **[site:mail]**:

Click a token to insert it into the field you've last clicked.		
NAME	TOKEN	DESCRIPTION
Array		Tokens related to arrays of strings.
Comments		Tokens for comments posted on the site.
Content types		Tokens related to content types.
Current date		Tokens related to the current date and time.
Current page		Tokens related to the current page request.
Current user		Tokens related to the currently logged in user.
Dates		Tokens related to times and dates.
Files		Tokens related to uploaded files.
Menu links		Tokens related to menu links.
Menus		Tokens related to menus.
Nodes		Tokens related to individual content items, or "nodes".
Random		Tokens related to random data.
Site information		Tokens for site-wide settings and other global information.
Email	[site:mail]	The administrative email address for the site.
Login page	[site:login-url]	The URL of the site's login page.
Name	[site:name]	The name of the site.
Slogan	[site:slogan]	The slogan of the site.
URL	[site:url]	The URL of the site's front page.
URL (brief)	[site:url-brief]	The URL of the site's front page without the protocol.
Taxonomy terms		Tokens related to taxonomy terms.
URL		Tokens related to URLs.
Users		Tokens related to individual user accounts.
Vocabularies		Tokens related to taxonomy vocabularies.

7. In the **Subject** field, enter **New user account registration on**.
8. Scroll down to the tokens table, and select the option **[site:name]** in the **Site information** section.
9. In the **Message** field, enter **The user [user:name], has just registered a new account on [site:name]:**

Label <input type="text" value="Send new user notification email to admin"/> A unique label for this advanced action. This label will be displayed in the interface of modules that integrate with actions, such as Trigger module.
Recipient <input type="text" value=" [site:mail]"/> The email address to which the message should be sent OR enter [node:author:mail], [comment:author:mail], etc. if you would like to send an e-mail to the author of the original post.
Subject <input type="text" value="New user account registration on [site:name]"/> The subject of the message.
Message <input type="text" value="The user [user:name], has just registered a new account on [site:name]"/> The message that should be sent. You may include placeholders like [node:title], [user:name], and [comment:body] to represent data that will be different each time message is sent. Not all placeholders will be available in all contexts.

10. Click on **Save**.
11. Select **Structure** from the admin menu, then select **Triggers**.
12. Select the **User** tab.
13. For the field **TRIGGER: AFTER CREATING A NEW USER ACCOUNT**, select the option **Send new user notification email to admin:**

TRIGGER: AFTER CREATING A NEW USER ACCOUNT	
<input type="text" value="Send new user notification email to admin"/>	<input type="button" value="Assign"/>

14. Click on **Assign**:

TRIGGER: AFTER CREATING A NEW USER ACCOUNT	
NAME	OPERATION
Send new user notification email to admin	unassign
<input type="button" value="Choose an action"/>	<input type="button" value="Assign"/>

How it works...

We begin by creating a new **Action**. In Drupal, actions are behaviors that can be activated by a selection of pre-defined Triggers, such as sending an e-mail, displaying a message, or blocking a user. We create a new action called **Send new user notification email to admin**, of the type **Send email**.

After giving the action its name in the **Label** field, we select the recipient e-mail address that we want the notification to be sent to. To do this, we insert the [site:mail] token by selecting it from the tokens lookup table. Using this token will mean that the notification will be sent to the site's current admin e-mail address. You can of course enter a static e-mail address in this field.

We then configure the **Subject** field so it has the following contents:

```
New user account registration on [site:name]
```

In this case we are using the token [site:name]. This will be replaced with the name of the site as set in the **Site information** settings.

Next, we enter the message that we want to appear in the e-mail:

```
The user [user:name], has just registered a new account on [site:name]
```

We use the token [user:name] to output the username that has registered for a new account, and we use the token [site:name] once more.

[ It is likely that you will want to customize your notification message further, and it's a good idea to experiment with the tokens that are available to provide the message that you require.]

After creating the action, we go to the **Triggers** page, where we configure the user trigger, **TRIGGER: AFTER CREATING A NEW USER ACCOUNT** to trigger the e-mail action that we have just created.

After completing all of the steps in the recipe, you can test the notification by registering a new user account. Upon successfully submitting an account registration, an e-mail will be delivered to the site admin's e-mail address.

There's more...

In this recipe we have seen how we generate a simple action from a pre-set trigger. However, what if you wanted something a bit more complex, with multiple triggers and conditions?

Building a notification system with the Rules module

The Rules module is a highly flexible tool that can be used to create outcomes which are triggered by multiple events, where you can set a range of conditions that need to be satisfied for the rule to proceed. A rule can have a series of actions, each of which can be carried out in a loop over a set of entities; for example, you could create a notification rule which sends an e-mail to each user with a particular role.

See also

- ▶ <http://drupal.org/project/rules>

Adding a biography field to the user profile

In its default configuration, the Drupal user profile features only a few fields such as Username, Password, E-mail, and Picture. There are many potential use cases where you would want to add more fields. Drupal makes it very easy for us to add additional fields to the user profile, and it's done in much the same way as you would edit a content type. In this recipe, we will be adding a new Long text field to the user profile to provide the user with the option of adding a biography.



Adding too many fields to a user profile can have performance implications. The user object is always loaded, and with it, any extra fields added to it.

Getting ready

To complete this recipe you will need to have completed the first recipe in this chapter, *Creating new user accounts*.

How to do it...

We will begin by creating a new **Biography** field for the user profile. We will then configure the field and set its position in the backend form, and finally, we will update an existing user account to add in some text for the **Biography** field:

1. Select **Configuration** from the admin menu, then select **Account settings**.
2. Select the **Manage fields** tab.
3. In the **Add new field** row, enter **Biography** in the **Label** field.
4. Enter **biography** in the **Field name** field.
5. Select **Long text** from the **FIELD** drop-down.
6. In the **WIDGET** column, select **Text area (multiple rows)**:

LABEL	NAME	FIELD	WIDGET	OPERATIONS
⊕ User name and password	account	User module account form elements.		
⊕ Timezone	timezone	User module timezone form element.		
⊕				
Add new field				
Biography	field_ biography	Long text	Text area (multiple rows)	
Label	Field name (a-z, 0-9, _)	Type of data to store.		Form element to edit the data.
⊕				
Add existing field				
	- Select an existing field -		- Select a widget -	
Label	Field to share			Form element to edit the data.

7. Click on **Save**.
8. In the **Field settings** popup, click on **Save field settings**.
9. On the following popup, enter the following text in the **Help text** field: **Use this field to enter an optional biography**.

10. For the **Text processing** field, select **Filtered text (user selects text format)**:

USER SETTINGS

These settings apply only to the *Biography* field when used in the *User* type.

Label *

Required field

Display on user registration form.
This is compulsory for 'required' fields.

Help text

Instructions to present to the user below this field on the editing form.
Allowed HTML tags: <a> <big> <code> <i> <ins> <pre> <q> <small> <sub> <sup> <tt> <p>

Text processing

Plain text

Filtered text (user selects text format)

11. Leave the remaining options in their default state and click on **Save settings**.

12. Drag the **Biography** row above the **Timezone** row using the drag handle:

LABEL	NAME	FIELD	WIDGET	OPERATIONS
⊕ User name and password	account	User module account form elements.		
⊕ Biography*	field_biography	Long text	Text area (multiple rows)	edit delete
⊕ Timezone	timezone	User module timezone form element.		
⊕				
Add new field	field_ <input type="text"/>	- Select a field type - <input type="button" value="▼"/>	- Select a widget - <input type="button" value="▼"/>	
Label	Field name (a-z, 0-9, _)	Type of data to store.	Form element to edit the data.	
⊕				
Add existing field		- Select an existing field - <input type="button" value="▼"/>	- Select a widget - <input type="button" value="▼"/>	
Label	Field to share	Form element to edit the data.		

Managing Users

13. Click on **Save**.
14. Select **People** from the admin menu, then find the user created in the recipe *Creating new user accounts*, **dummyuser1**.
15. Select **edit** for **dummyuser1**.
16. In the **Biography** field, enter some sample biography text:

The screenshot shows a text area with the heading "Biography". Inside the area, there is placeholder text: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis lectus metus, at posuere neque. Sed pharetra nibh eget orci convallis at posuere leo convallis. Sed blandit augue vitae augue scelerisque bibendum. Vivamus sit amet libero turpis, non venenatis urna." Below the text area, there is a "Text format" dropdown set to "Filtered HTML". To the right of the dropdown is a link "More information about text formats". Underneath the dropdown, there is a list of allowed HTML tags: "Web page addresses and e-mail addresses turn into links automatically.", "Allowed HTML tags: <a> <cite> <blockquote> <code> <dl> <dt> <dd>", and "Lines and paragraphs break automatically."

17. Click on **Save**.
18. Select **dummyuser1** from the list of users to view the frontend profile:

The screenshot shows the user profile for "dummyuser1". At the top, there is a title "dummyuser1" and a navigation bar with buttons for "View", "Edit", "Shortcuts", and "Devel". Below the title, there is a section titled "Biography:" containing the same sample text as the editor. There is also a "History" link. At the bottom, there is a "Member for" section showing "1 week 3 hours".

How it works...

We begin by opening up the **Account settings** page from where we go to the **Manage fields** tab. The **Manage fields** page for the user account is very similar to the **Manage fields** page for custom content types.

Next, we add the new **Biography** field. We enter the **Label** and **Field** name, and then choose the **Long text** field type. This field type allows users to enter large amounts of text using the WYSIWYG editor, if it's installed.

On the following configuration page, we then add some **Help text**, which serves as a reminder to the user what they can do with the field. Next, we set the **Text processing** field to **Filtered text**. This option will help to sanitize the user's input by filtering out undesirable input. This option will also allow the user to select from a list of available input methods, which will depend upon their permissions.

After saving the new field, we rearrange the order of the fields so that the **Biography** field appears before the **Timezone** field; we do this simply to move the **Biography** field to a more prominent position on the form, as it's more likely to be changed than the **Timezone**.

After saving the changes to the **Manage fields** page, you can now edit a user profile and add in some **Biography** text. This text will then be displayed on the user profile frontend display.

There's more...

We have seen a simple example of how the user profile can be extended, but there are almost endless possibilities when it comes to customizing the user profile. However, it's important to be aware that adding too many fields to the profile can cause performance issues. It may be preferable to consider the use of Profile 2 for adding extra fields.

Adding other fields

We have seen how to add a biography field to the user profile. However, there are other fields that you might want to add such as Date of birth, which you could set using the Date module, or a field to store the user's gender.

Using Views to build a biographies page

After adding the **Biography** field, it's now possible to create a View that displays a list of users next to their biographies.

Profile 2

The Profile 2 module can provide a more-powerful means of controlling what is displayed on a user profile. It also allows you to extend the range of user profile fields, without adding to the core user object that's loaded on every page load.

- ▶ <http://drupal.org/project/profile2>

Building a grid view of profile pictures

By default, in Drupal, there is no default output of the users registered on the site. In this recipe, we will create a staff-listing page which displays the profile pictures of all staff members above their names and job titles.

Getting ready

Before starting this recipe you will need to install the latest recommended releases of the following two modules:

- ▶ <http://drupal.org/project/views>
- ▶ <http://drupal.org/project/ctools>

Once you have installed these modules, enable the following features:

- ▶ Chaos tools
- ▶ Views
- ▶ Views UI

How to do it...

In this recipe, we will first add some extra fields to the user profile; we will also be adding a new Employee user role. We will then make a View to display the staff members in a grid form, and finally we will add some test users to demonstrate the output.

Our Team

 Dylan James Web developer	 Janice James Sculptor	 Julian James Actor	 Kristian Humphrey 3D artist
 Lauren James Animator			

1. Select **Configuration** from the admin menu, then select **Account settings**.
2. Select the **Manage fields** tab.

3. In the **Add new field** row, enter **Job title**.
4. In the **Field name** field, enter **job_title**.
5. In the **FIELD** drop-down, select **Text**:

LABEL	NAME	FIELD	WIDGET	OPERATIONS
⊕ User name and password	account	User module account form elements.		
⊕ Timezone	timezone	User module timezone form element.		
⊕ Add new field				
Job title	field_job_title	Text	Text field	
Label	Field name (a-z, 0-9, _)	Type of data to store.		Form element to edit the data.
⊕ Add existing field				
	- Select an existing field -		- Select a widget -	
Label	Field to share			Form element to edit the data.

6. Click on **Save**.
7. In the **Field settings** popup, leave the **Maximum length** field set as **255**, then click on **Save field settings**.
8. For the **Text processing** field, select the option **Plain text**:

USER SETTINGS
These settings apply only to the *Job title* field when used in the *User* type.

Label *

Required field

Display on user registration form.
This is compulsory for 'required' fields.

Help text

Instructions to present to the user below this field on the editing form.
Allowed HTML tags: <a> <big> <code> <i> <ins> <pre> <q> <small> <sub><sup> <tt> <p>

Text processing
 Plain text
 Filtered text (user selects text format)

Managing Users

9. Leave all other options in their default states, then click on **Save settings**. Select **People** from the admin menu, then select **Permissions**.
10. Select the **Permissions** tab, then select the **Roles** secondary tab below.
11. Enter **Employee** in the add new row field:

NAME	OPERATIONS	
+ anonymous user (<i>locked</i>)		edit permissions
+ authenticated user (<i>locked</i>)		edit permissions
+ administrator	edit role	edit permissions
+ Editor	edit role	edit permissions
Employee		Add role

12. Click on **Add role**.
13. Select **Structure** from the admin menu, then select **Views**.
14. Select **+Add new view**.
15. In the **View name** field, enter **Staff listing**.
16. Configure the options in the first fieldset to read as follows: **Show Users sorted by Unsorted**.
17. In the **Create a page** fieldset, ensure that **Create a page** is checked.
18. In the **Page title** field, enter **Our Team**.
19. In the **Path** field, enter **our-team**.
20. Select **Grid** from the **Display format** drop-down.
21. In the following field, ensure that **Fields** is selected, so that the **Display format** configuration reads **Grid of Fields**.
22. Set the **Items to display** field to **16**.
23. Ensure that **Use a pager** is checked.
24. Check the option **Create a menu link**.
25. Select **Main menu**, from the **Menu** field.

26. Enter **Our Team** in the **Link text** field:

View name *
Staff listing Machine name: staff_listing [Edit]

Description

Show sorted by

Create a page

Page title

Path

Display format
 of

Items to display

Use a pager

Create a menu link

Menu

Link text

27. Click on **Continue & edit**.

28. In the **Fields** section, click on the **Add** button.

29. In the **Search** field, enter the term **picture** to narrow down the list of search results.

30. Check the option **User: Picture**:

Add fields

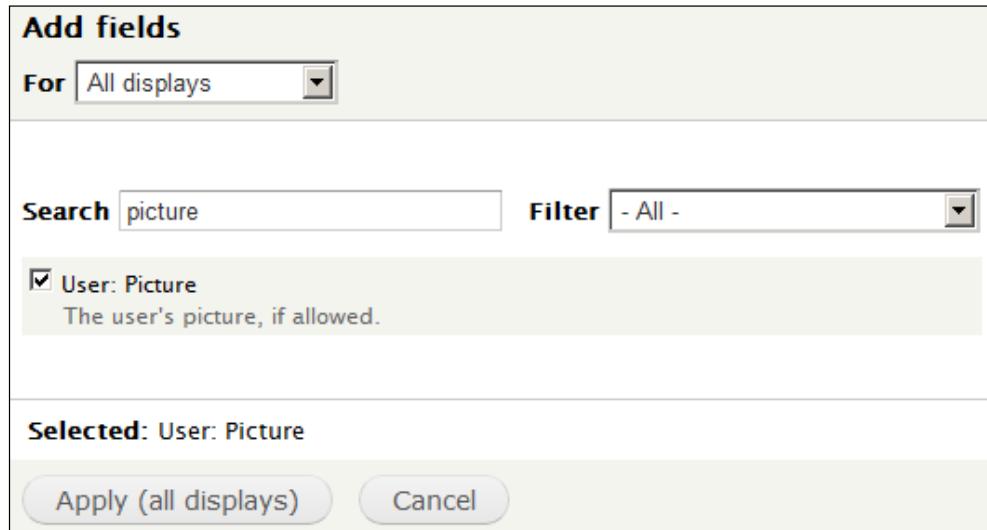
For All displays

Search picture **Filter** - All -

User: Picture
The user's picture, if allowed.

Selected: User: Picture

Apply (all displays) **Cancel**



31. Select **Apply (all displays)**.
32. In the **Configure field** popup, un-check the option **Create a label**.
33. In the **Image style** drop-down, select the option, **thumbnail**:

The user's picture, if allowed.

Create a label
Enable to create a label for this field.

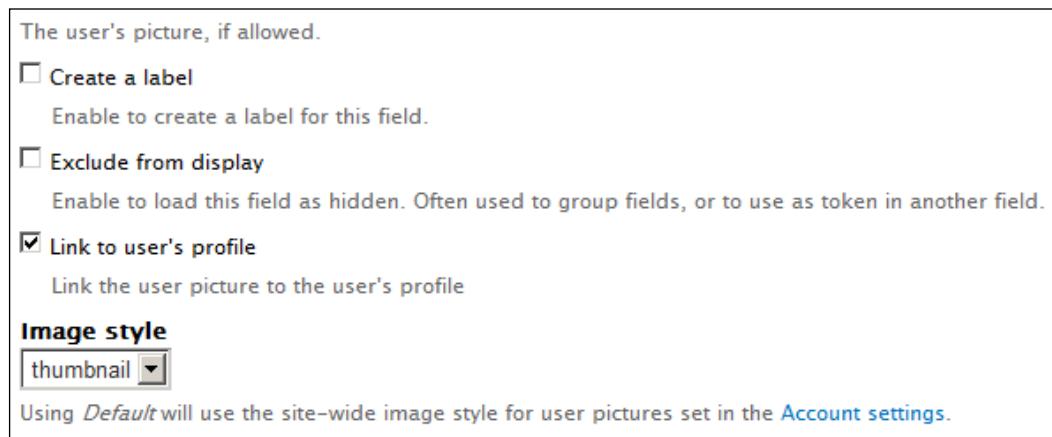
Exclude from display
Enable to load this field as hidden. Often used to group fields, or to use as token in another field.

Link to user's profile
Link the user picture to the user's profile

Image style

thumbnail

Using **Default** will use the site-wide image style for user pictures set in the [Account settings](#).



34. Select **Apply (all displays)**.
35. Still in the **Fields** section, click on the **Add** button.
36. In the **Search** field, enter the term **Job** to narrow down the list of search results.

37. Check the option **User: Job title**:

Add fields

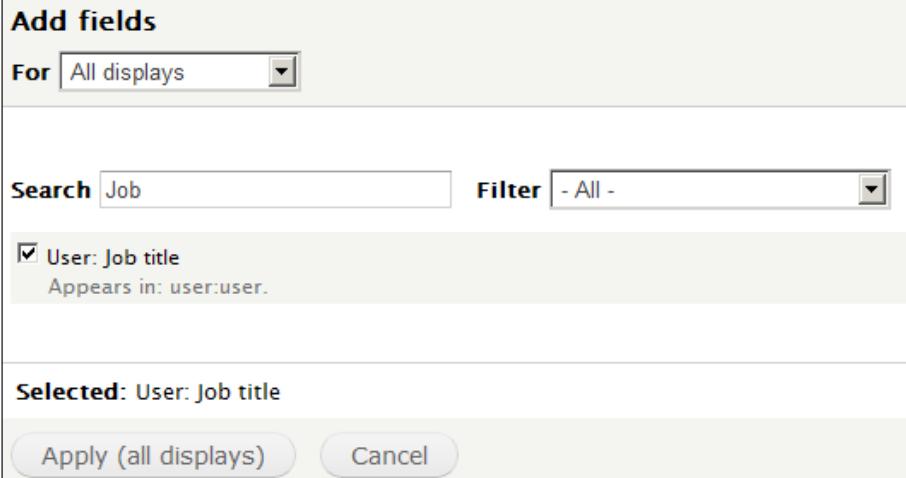
For All displays

Search Job **Filter** - All -

User: Job title
Appears in: user:user.

Selected: User: Job title

Apply (all displays) **Cancel**



38. Select **Apply (all displays)**.

39. In the **Configure field** popup, un-check the option **Create a label**.

40. Set the **Formatter** field to **Plain text**:

Configure field: User: Job title

For All displays

Appears in: user:user.

Create a label
Enable to create a label for this field.

Exclude from display
Enable to load this field as hidden. Often used to group fields, or to use as token in another field.

Formatter

Plain text



41. Select **Apply (all displays)**.

42. Select the drop-down button next to the **add** button in the **Fields** section, then choose the option **Rearrange**.

43. Drag the row **User: Name** below **User: Picture**:

		REMOVE
⊕	User: Picture	Remove
⊕	User: Name *	Remove
⊕	User: Job title	Remove

44. Select **Apply (all displays)**.

45. In the **Filter criteria** section, click on the **add** button.

46. In the **Search** field, enter the term **Role** to narrow down the list of search results.

47. Check the option **User: Roles**:

Add filter criteria

For All displays

Search Role **Filter** - All -

User: Roles
Roles that a user belongs to.

Selected: User: Roles

[Apply \(all displays\)](#) [Cancel](#)

48. Select **Apply (all displays)**.

49. In the **Configure filter criterion** popup, ensure that the **Operator** field is set to **Is one of**.

50. In the **Options** field, select **Employee**:

Configure filter criterion: User: Roles

For All displays

Roles that a user belongs to.

Expose this filter to visitors, to allow them to change it

Operator	Options
<input checked="" type="radio"/> Is one of	Select all
<input type="radio"/> Is all of	administrator
<input type="radio"/> Is none of	Editor
<input type="radio"/> Only has the 'authenticated user' role	Employee
<input type="radio"/> Has roles in addition to 'authenticated user'	

Reduce duplicates

This filter can cause items that have more than one of the selected options to appear as duplicate results. If this filter causes duplicate results to occur, this checkbox can reduce those duplicates; however, the more terms it has to search for, the less performant the query will be, so use this with caution. Shouldn't be set on single-value fields, as it may cause values to disappear from display, if used on an incompatible field.

51. Select **Apply (all displays)**.
52. In the **Sort criteria** section, click on the **add** button.
53. In the **Search** field, enter the term **Name**, to narrow down the list of search results.
54. Check the option **User: Name**:

Add sort criteria

For All displays

Search Name Filter - All -

User: Name
The user or author name.

Selected: User: Name

Apply (all displays) Cancel

Managing Users

55. Select **Apply (all displays)**.
56. In the **Configure sort criterion** popup, leave the option **Sort ascending** selected and select **Apply (all displays)**:

The screenshot shows the 'Page details' configuration screen. At the top, there are buttons for 'Page*' (selected), '+ Add', and 'edit view name/description'. Below that is a 'view page' button. The main area is divided into several sections:

- TITLE**: Title: Our Team
- FORMAT**: Format: Grid | Settings
- FIELDS**: User: Name, User: Picture, User: Job title
- FILTER CRITERIA**: User: Active (Yes), User: Roles (= Employee)
- SORT CRITERIA**: User: Name (asc)
- PAGE SETTINGS**: Path: /our-team, Menu: Normal: Our Team, Access: Permission | View user profiles
- HEADER** and **FOOTER** sections with 'add' buttons
- PAGER**: Use pager: Full | Paged, 16 items, More link: No

57. Click on **Save**.
58. Select **People** from the admin menu, then select **+Add user**.
59. In the **Username** field, enter a sample **Test employee** (or other suitable username).
60. In the **E-mail address** field, enter **testemployee@example.com** (or other suitable e-mail address).
61. Enter **password** in the **Password** and **Confirm password** fields (or other suitable password).
62. Ensure that **Active** is selected for the **Status** field.

63. In the **Roles** section, check the **Employee** option:

Username *
 Spaces are allowed; punctuation is not allowed except for periods, hyphens, apostrophes, and underscores.

E-mail address *
 A valid e-mail address. All e-mails from the system will be sent to this address. The e-mail address is not made public and will only be used if you wish to receive a new password or wish to receive certain news or notifications by e-mail.

Password *
 Password strength: Fair

Confirm password *
 Passwords match: yes

To make your password stronger:
• Add uppercase letters
• Add numbers
• Add punctuation

Provide a password for the new account in both fields.

Status
 Blocked
 Active

Roles
 authenticated user
 administrator
 Editor
 Employee

64. Click on **Create new account**.
65. Select **People** from the admin menu, then select **edit** for the newly created user **Test employee**.
67. In the **Picture** section, click on **Browse**.
68. Find a suitable picture to use from your computer then click on **Open**.
69. In the **Job title** field, enter **Web developer** (or other suitable job title).
70. Click on **Save**.

71. Add some more users of your choice as in steps 65 to 70 for demonstration purposes.
72. Go to the URL `Drupal root /our-team` to see the listing of employees.

How it works...

We start by editing the user profile fields. We add a new Text field called **Job title** and configure it to have a maximum length of 255 characters.

We add a new **Role** called **Employee**, which we are going to use to differentiate between regular site users and employees so that we can filter our view to display only users that have been assigned the Employee role.

Next, we create a new **View** named **Staff listing** which will store our page display, **Our Team**. We set the **Display** format to **Grid**. This formats the layout of the display into a grid form using tables.

Following the configuration of the view setup screen we next begin to add some more fields to the display. First we add the **User: Picture**. We disable the **Label** field and set the **Image** style to **Thumbnail**. You may want to create a new **Image style** to use here, depending on the size you would like the image to display.

After adding the **Picture** field, we add the **User: Job title** field, for which we hide the label and set the Formatter to **Plain text**, so that the **Job title** output doesn't link to anything.

After adding the fields, we rearrange them so that the **Picture** is on the top, followed by the **Username**, then the **Job title**.

Next, we add a filter using the **Role** field. We configure the filter to require that the user has the **Employee** role to ensure that we don't display regular users on the **Our Team** display.

After configuring the filter, we add a sort criterion to sort the users **alphabetically** on the username.

Finally, we save the view and create a sample user to test the view. After creating the user, we go back and edit the account because some of the fields are unavailable when first creating an account.

12

Running Drupal

In this chapter we will cover:

- ▶ System maintenance
- ▶ Setting up a backup system
- ▶ Search Engine Optimization (SEO) with Drupal
- ▶ Securing a Drupal installation
- ▶ Configuring Drupal caching
- ▶ Running commands with the Drush tool

Introduction

This chapter is about the on-going tasks associated with running a Drupal site and tuning it for optimal performance.

In the recipe *System maintenance*, we will see how to keep the site well maintained, by regularly updating the modules, and monitoring the status of the site.

An essential consideration for any website is its backup procedure. In the recipe *Setting up a backup system*, we will create an on-going and automated backup system, which safely backs up the database and user files.

Running a successful website is also about the quality of the content. In the recipe *Search Engine Optimization (SEO) with Drupal*, we will see how to configure an SEO checklist, which will guide you through optimizing your site for search engines.

Security is another important consideration for any site. In the recipe, *Securing a Drupal installation*, we will configure a Security checklist module to highlight any of the site's serious security vulnerabilities.

We will see how to improve the site's performance in the recipe *Configuring Drupal caching*, where we will configure Drupal's built-in caching, Panel caching, and finally, Views caching. Configuring just these three areas can achieve significant performance boosts.

Finally, we delve into the world of the Drush project in the recipe *Running commands with the Drush tool*. Drush is a command-line tool that will literally save hours of your time by letting you install and enable modules in a few seconds, and many other functions. In this recipe, we will configure the Drush tool, and run through some of its most useful commands.

System maintenance

To keep a Drupal website well maintained, it's important that it's monitored regularly, and kept up to date. This means that you need to be aware when updates are available on your site, especially security updates. You will also need to be aware of any other issues that may arise.

This recipe will explain how to update the modules on your site, and to check through the system reports for any possible issues.

Getting ready

In this recipe we will be updating the site's modules, so you will need to have just received an e-mail alert announcing that there are new releases for modules on your site. Alternatively, you could install a previous version of a module, and run the update checker to activate the "New releases available" alert.



Before performing an update of the Drupal site, ensure that you run a manual backup so that you can recover your database in the event that any of the updates introduce a conflict or an error. See the *Setting up a backup system* recipe of this chapter.

How to do it...

In the first part of this recipe, we will be updating the Drupal modules to the latest versions; we will then be checking the status report for any issues, and finally, we will check the logs for any serious issues:

1. Select **Reports** from the admin menu, and then select **Available updates**.
2. Select the **UPDATE** tab.
3. Check all of the boxes for the available updates:

<input checked="" type="checkbox"/>	NAME	INSTALLED VERSION	RECOMMENDED VERSION
<input checked="" type="checkbox"/>	Panels (Security update)	7.x-3.0-alpha3	7.x-3.2 (Release notes)
<input checked="" type="checkbox"/>	Chaos tool suite (ctools) (Security update)	7.x-1.0-rc1	7.x-1.0 (Release notes)
<input checked="" type="checkbox"/>	Date	7.x-2.0-rc1	7.x-2.3 (Release notes)
<input checked="" type="checkbox"/>	Forum Access	7.x-1.0-beta1	7.x-1.0 (Release notes)
<input checked="" type="checkbox"/>	Menu block	7.x-2.2	7.x-2.3 (Release notes)
<input checked="" type="checkbox"/>	Views	7.x-3.0	7.x-3.3 (Release notes)

4. Click on **Download these updates**.
5. On the **Ready to update** pop up, ensure that the checkbox **Perform updates with site in maintenance mode** is checked.
6. Click on **Continue**.

7. On the following page, select the link, **Run database updates**:

The screenshot shows a list of modules and their installation status:

- panels**: Installed *panels* successfully
- ctools**: Installed *ctools* successfully
- date**: Installed *date* successfully
- forum_access**: Installed *forum_access* successfully
- menu_block**: Installed *menu_block* successfully
- views**: Installed *views* successfully

Next steps

- Your modules have been downloaded and updated.
- [Run database updates](#)

8. If all of the requirements are satisfied, you will see **Database update page**. Click on **Continue**; otherwise, fix the requirement issues first.
9. On the following page, select **Apply pending updates**.
10. After the updates have been completed, select the link to **Front page**.
11. Select **Reports** from the admin menu, and then select **Status report**.

12. Check through the report for any problem areas (highlighted in red), addressing issues as necessary:



The screenshot shows the Drupal Status report page. At the top, there's a navigation bar with links for Dashboard, Content, Structure, Appearance, People, Modules, Configuration, Reports (which is highlighted in blue), and Help. The user is logged in as 'Hello dylan'.

The main content area is titled 'Status report'. It contains a table with various system parameters and their current status. Some rows have a light blue background, while others have a light green background. A few specific rows are highlighted in red, indicating potential issues:

Drupal		7.12
Access to update.php	Protected	
Configuration file	Protected	
Cron maintenance tasks	Last run 15 sec ago	
You can run cron manually .		
To run cron from outside the site, go to http://[REDACTED]/cron.php?cron_key=eXKSFT5Ux1_aCRGm6JEAIRAr55RTifOtxemXJpcNjBU		
Database system	MySQL, MariaDB, or equivalent	
Database system version	5.1.56	
Database updates	Up to date	
Drupal core update status	Up to date	
File system	Writable (<i>public</i> download method)	
GD library PNG support	bundled (2.0.34 compatible)	
GD library rotate and desaturate effects	bundled (2.0.34 compatible)	
Module and theme update status	Up to date	
Node Access Permissions	Disabled	
If the site is experiencing problems with permissions to content, you may have to rebuild the permissions cache. Rebuilding will remove all privileges to content and replace them with permissions based on the current modules and settings.		
Rebuilding may take some time if there is a lot of content or complex permission settings. After rebuilding has completed, content will automatically use the new permissions. Rebuild permissions		
PHP	5.2.17 (more information)	
PHP extensions	Enabled	
PHP memory limit	124M	
PHP register globals	Disabled	
Unicode library	PHP Mbstring Extension	
Update notifications	Enabled	
Upload progress	Not enabled	
Your server is capable of displaying file upload progress, but does not have the required libraries. It is recommended to install the PECL uploadprogress library (preferred) or to install APC .		
Web server	Apache	

13. Select **Reports** from the admin menu, and then select **Recent log messages**.
14. For the **Severity** field, select the values **Emergency**, **Alert**, **Critical**, and **Error**, by holding down the *Ctrl* key, then select **Filter**.
15. Address any log issues that are discovered.

How it works...

To begin, we go to the **Available updates** page, and then to the **Update** page. The **Update** page allows you to select which updates to apply, and the Drupal updater will do the rest. We choose to apply all of the available updates. After the modules have downloaded, Drupal needs to update the modules on the system. We choose to do this in **maintenance mode**, to prevent users seeing any glitches.

After the updates have been applied, we select the link **Run database updates**. This applies any required modifications to the modules' database tables. Before the database updates are implemented, the updater will check that all of the requirements are met. Sometimes, the current version of the Drupal core will not be high enough for one of the updated modules, in which case, the core will need to be updated. If all requirements are met, then we select the option **Apply pending updates**.

After implementing the updates to the site's modules, we go to the **Status report**. The status report gives a high-level overview of any issues on the site; it also displays some important information about the site, such as the current Drupal version, the server's PHP version, and the status of the Cron. It's very important to spend some time familiarizing yourself with the information on this page.

In the final segment of this recipe, we take a look at the logs. We filter the logs to show only the most important issues, but it's a good idea to periodically check all logs, especially if you are trying to solve an issue with a malfunctioning module.

Setting up a backup system

Configuring a backup system is essential for even the smallest of websites. This recipe is about creating a simple on-going backup system to make regular and automated backups of the content of a site, that is to say, its database and user files. This recipe does not specify a system for backing up the Drupal core files, custom themes, and contributed modules. These must be managed separately.

Getting ready

Before beginning this recipe, please install and enable the most recent releases of the following two modules:

- ▶ http://drupal.org/project/backup_migrate.
- ▶ http://drupal.org/project/backup_migrate_files.

To install the Backup and Migrate Files module, you will need to enable the `PEAR Archive_Tar` library.

How to do it...

We will start the recipe by configuring the site's private folder. We will then create backup profiles and backup schedules for each of the three items we are going to back up—the site's database, its public files, and its private files:

1. Select **Configuration** from the admin menu, then select **File system**.
2. In **Private file system path**, enter: `sites/default/files/private`.
3. Click on **Save configuration**.
4. Select **Configuration** from the admin menu, then select **Backup and Migrate**.
5. Select the **PROFILES** tab.
6. Select **+Add profile**.
7. In the **Profile Name** field, enter **Database backup**.
8. In the **Backup file name** field, add the text **-database** to the field, so that it reads **[site:name]-database**:

The screenshot shows the 'Database backup' profile configuration page. The 'Profile Name' field contains 'Database backup'. Under the 'BACKUP FILE' section, the 'Backup file name' field contains '[site:name]-database'. A note below says 'You can use tokens in the file name.' and there is a checked checkbox for 'Append a timestamp'. The 'Timestamp format' field contains 'Y-m-d\TH-i-s'. A note below says 'Should be a PHP date() format string.' Under 'File Encryption', it says 'Install the AES Encryption Module to enable backup file encryption.' The 'Compression' dropdown is set to 'GZip'.

9. Leave the other options as they are, and click on **Save profile**:

NAME	SOURCE	FILENAME	OPERATIONS
Default Settings	Default Database	[site:name]	override
Database backup	Default Database	[site:name]-database	edit delete

10. Select the **SCHEDULES** tab.
11. Select **+Add Schedule**.
12. In the **Schedule Name** field, enter **Database backup**.
13. In the **Backup Source** field, ensure that **Default Database** is selected.
14. For the **Settings Profile** field, select the **Database backup** option.
15. Enter **12** in the **Backup every** field, and select the **Hours** option.
16. In the **Number of Backup files to keep** field, enter **14**.
17. For the **Destination** field, ensure that **Scheduled Backups Directory** is selected:

Schedule Name
Database backup

▼ BACKUP SOURCE

Backup Source
 Choose the database to backup. Any database destinations you have created and any databases specified in your settings.php can be backed up.

Settings Profile
 [Create new profile](#)

Backup every The number of backup files to keep before deleting old ones. Use 0 to never delete backups. **Other files in the destination directory will get deleted if you specify a limit.**

Number of Backup files to keep

Destination
 Choose where the backup file will be saved. Backup files contain sensitive data, so be careful where you save them. [Create new destination](#)

18. Click on **Save schedule**:

NAME	DESTINATION	PROFILE	FREQUENCY	KEEP	ENABLED	LAST RUN	OPERATIONS
Database Directory	Scheduled Backups Database backup	Database backup	Every 12 hours	14	Enabled	Never	edit delete

19. Select the **PROFILES** tab.20. Select **+Add Profile**.21. In the **Profile Name** field, enter **Public files backup**.22. In the **Backup file name** field, add the text **-public_files**, so that it reads **[site:name]-public_files**:

Profile Name *
Public files backup

▼ BACKUP FILE

Backup file name
[site:name]-public_files
You can use tokens in the file name.

Append a timestamp.

Timestamp format
Y-m-d\TH-i-s
Should be a PHP [date\(\)](#) format string.

Compression
GZip

File Encryption
Install the [AES Encryption Module](#) to enable backup file encryption.

23. Leave the other options as they are, and then click on **Save profile**:

NAME	SOURCE	FILENAME	OPERATIONS
Default Settings	Default Database	[site:name]	override
Database backup	Default Database	[site:name]-database	edit delete
Public files backup	Default Database	[site:name]-public_files	edit delete

24. Select the **SCHEDULES** tab.

25. Select **+Add Schedule**.

26. In the **Schedule Name** field, enter **Public files backup**.

27. For the **Backup Source** field, select **Public Files Directory**.

28. For the **Settings Profile** field, select **Public files backup**.

29. Enter **12** in the **Backup every** field and select the **Hours** option.

30. In the **Number of Backup files to keep** field, enter **6**.

31. For the **Destination** field, ensure that **Scheduled Backups Directory** is selected:

Schedule Name
Public files backup

▼ BACKUP SOURCE

Backup Source
Public Files Directory

Choose the database to backup. Any database destinations you have created and any databases specified in your settings.php can be backed up.

Settings Profile
Public files backup

[Create new profile](#)

Backup every 12 Hours

Number of Backup files to keep
6

The number of backup files to keep before deleting old ones. Use 0 to never delete backups. **Other files in the destination directory will get deleted if you specify a limit.**

Destination
Scheduled Backups Directory

Choose where the backup file will be saved. Backup files contain sensitive data, so be careful where you save them. [Create new destination](#)

32. Click on **Save schedule**:

NAME	DESTINATION	PROFILE	FREQUENCY	KEEP	ENABLED	LAST RUN	OPERATIONS
Database backup	Scheduled Backups Directory	Database backup	Every 12 hours	14	Enabled	Never	edit delete
Public files backup	Scheduled Backups Directory	Public files backup	Every 12 hours	6	Enabled	Never	edit delete

33. Select the **PROFILES** tab.

34. Select **+Add profile**.

35. In the **Profile Name** field, enter **Private files backup**.

36. In the **Backup file name** field, add the text **-private_files**, so that it reads **[site:name]-private_files**:

Profile Name *
Private files backup

▼ BACKUP FILE

Backup file name
[site:name]-private_files
You can use tokens in the file name.
 Append a timestamp.

Timestamp format
Y-m-d\TH-i-s
Should be a PHP [date\(\)](#) format string.

Compression
GZip

File Encryption
Install the [AES Encryption Module](#) to enable backup file encryption.

37. Leave the other options as they are, and then click on **Save profile**:

NAME	SOURCE	FILENAME	OPERATIONS
Default Settings	Default Database	[site:name]	override
Database backup	Default Database	[site:name]-database	edit delete
Public files backup	Default Database	[site:name]-public_files	edit delete
Private files backup	Default Database	[site:name]-private_files	edit delete

38. Select the **SCHEDULES** tab.
39. Select **+Add Schedule**.
40. In the **Schedule Name** field, enter **Private files backup**.
41. For the **Backup Source** field, select **Private Files Directory**.
42. For the **Settings Profile** field, select **Private files backup**.
43. Enter **12** in the **Backup every** field and select the **Hours** option.
44. In the **Number of backups** to keep field, enter **6**.
45. For the **Destination** field, ensure that **Scheduled Backups Directory** is selected:

Schedule Name
Private files backup

▼ BACKUP SOURCE

Backup Source
Private Files Directory

Choose the database to backup. Any database destinations you have created and any databases specified in your settings.php can be backed up.

Settings Profile
Default Settings

[Create new profile](#)

Backup every 12

Number of Backup files to keep
6

The number of backup files to keep before deleting old ones. Use 0 to never delete backups. Other files in the destination directory will get deleted if you specify a limit.

Destination
Scheduled Backups Directory

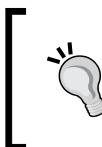
Choose where the backup file will be saved. Backup files contain sensitive data, so be careful where you save them.
[Create new destination](#)

46. Click on **Save schedule**:

NAME	DESTINATION	PROFILE	FREQUENCY	KEEP	ENABLED	LAST RUN	OPERATIONS
Private files backup	Scheduled Backups Directory	Default Settings	Every 12 hours	6	Enabled	Never	edit delete
Database backup	Scheduled Backups Directory	Database backup	Every 12 hours	14	Enabled	Never	edit delete
Public files backup	Scheduled Backups Directory	Public files backup	Every 12 hours	6	Enabled	Never	edit delete

How it works...

Before configuring the scheduled backups, we need a secure location to store them. In this recipe, we use a destination on the server. In order for this to be inaccessible over the Web, we need to ensure that **Private file system path** is configured. In Drupal, this location is intended for the storage of all private files, which are not for direct access over the Web, and when the location is specified, Drupal places a `.htaccess` file in the directory to prevent direct access.



It's important to test whether the files uploaded to the private directory are actually private, and inaccessible through direct access. It's also important to note that non-Apache servers may require further configuration.

Once we have configured our secure-file directory, we begin to configure the backup system.

We are making backups of three distinct items:

- ▶ The site's database
- ▶ The public files directory—all content files uploaded to the site, into the publicly accessible directory
- ▶ The private files directory—all content files uploaded to the site, into the private directory; only accessible by a Drupal-generated link, depending on the user's permissions

To make these backups, we need to first set up a backup profile, and then a backup schedule.

We create a new backup profile, which we call **Database backup**. We configure **Backup file name** so that the backup file is distinguishable from other types of backups. The filename already contains a token, which outputs the site name, as set in the Site information page. We add the text **-database**. We also leave the default option **Append a timestamp** enabled, so that the filename contains the exact date and time of the backup (essential when you have multiple backups in one day). So as an example, filename output by this profile could be as follows:

```
MyDrupalsite-database-2012-04-06T10-15-40.mysql.gz
```

After creating the profile, we create the database backup schedule. The schedule determines how often a particular backup is run.

We set **Schedule name**, to **Database backup**, then we set **Backup source** to **Default database**. The **Backup source** field is where we select what is being backed up in this schedule. There are only three options, and these are determined by the module. We set the **Settings profile** to the profile we created, **Database backup**. In the **Backup every** field, we specify that we want this schedule to run every **12 hours**. We then specify that we want to keep 14 backups, which equates to a week's worth. The final backup schedule option is the **Destination** setting. This currently only has one available option, **Scheduled backups directory**, which will create the backup inside a `backup_migrate` folder, inside the private folder that we created earlier on in the recipe.



The frequency of a backup schedule is highly dependent on factors such as hosting space, bandwidth allowance, and frequency of change of content. In an ideal world, you would back up your site at least every hour, but this is most likely completely impractical when your backup files become larger. You will need to tailor these settings to match your needs and your server. It might be easier to ask the question, "How much data am I prepared to lose in the event of failure?".

We then move onto the public files backup. Similarly to the database backup, we create a new profile for the public files backup, but we set the **Backup file name** to **[site:name]-public_files**. This will create backup filenames in the following form:

`MyDrupalSite-public_files-2012-04-06T12-23-49.tar.gz`

We then create a **Public files backup** schedule, where we set **Backup source** to **Public files directory**, using the **Public files backup** profile. We set the backup frequency to every **12 hours**, but we set **Number of backups to keep** to **6**. This is because the backup files will take up much more space than the database, so we keep only three days of backups. Feel free to increase this number if you calculate that you have ample disk space to accommodate the backups.

Next, we configure the private files backup. This is very similar to the previous two configurations. We create a new backup profile where we set the **Backup file name** to **[site:name]-private_files**. This will create backup filenames in the following form:

`MyDrupalSite-private_files-2012-04-06T12-25-04.tar.gz`

We then create a **Private files backup** schedule, where we set **Backup source** to **Private files directory**, using the **Private files backup** profile. We set the backup frequency to every **12 hours**, and again, we set **Number of backups to keep** to **6**.



This recipe only concerns the backing up of the database, and the user files, that is to say, all files uploaded through the site, such as content images, documents, and videos. It's very important to note that backups of the Drupal core, site settings, the theme files, and the contributed modules used on the site are not included. To back up these files, you should ensure that your hosting company keeps regular backups of your site. You should also ensure that each time you add a new module, or modify a theme, you download your site and create a backup of it. Preferably, you should store these files in a Source control system, such as SVN or Visual SourceSafe.

The backup of the themes, modules, and Drupal core, is treated differently from the backup of the database and user files because once the site is live, they should not change very often.

There's more...

We have seen how to create a simple backup system of the database and user files, but we haven't discussed restoring from a backup file. Also, there are alternative backup destinations that should be considered, in addition to the server backup.

Restoring from a backup file

There will sometimes be situations where your site becomes broken, or some data is accidentally deleted. In these situations you can restore the database or files from the **Restore** tab of the **Backup and Migrate** module by uploading a file from your local system. Alternatively, you can view the list of backup files on the sever through the **Destinations** tab, and restore one of those backup files directly through its **Restore** option.

Creating manual backups

It's very prudent to create a manual backup of the database when carrying out a major operation, such as an upgrade on the site or a menu restructuring. You can do this through the **Backup** tab of the Backup and Restore module. From this page you can choose to back up any of the three backup items (database, public files, and private files), to any of your designated destinations, and also download the file directly.

Using alternative backup destinations

In this recipe, we have configured only one backup destination, which is the server where the website sits. There are a range of other backup destinations that can be configured, and it's worth considering them in addition to your web server backup, to reduce the risk of data loss.

The other destinations that can be configured are as follows:

- ▶ FTP directory: This can be any secure FTP location that you have access to
- ▶ The Amazon S3 bucket
- ▶ E-mail

See also

- ▶ Drupal Backup and migrate module: http://drupal.org/project/backup_migrate
- ▶ Backing up a site: <http://drupal.org/node/22281>

Search Engine Optimization (SEO) with Drupal

SEO in itself is a very large subject area; however, in this recipe we will cover some of the most effective methods of optimizing your site within Drupal. We will see how to configure custom page titles, bespoke URLs, and how to generate an XML sitemap and submit it to the search engines.

Getting ready

To complete this recipe, install the following modules:

- ▶ <http://drupal.org/project/token>
- ▶ http://drupal.org/project/page_title
- ▶ <http://drupal.org/project/pathauto>
- ▶ <http://drupal.org/project/xmlsitemap>

Once installed, enable the following features of the modules:

- ▶ Pathauto
- ▶ Token
- ▶ Page title
- ▶ XML sitemap
- ▶ XML sitemap engines
- ▶ XML sitemap menu
- ▶ XML sitemap node

How to do it...

We will start by editing the Article content type to enable custom page titles, and inclusion in the XML sitemap. We will then optimize the URLs that are generated for the Article type, and finally, we will configure the XML sitemap:

1. Select **Structure** from the admin menu, then select **Content types**.
2. Click on **Edit** for the **Article** content type.
3. Select the **Page Title Settings** tab.
4. Check the **Show field** option.

Submission form settings Title	Page Title Field <input checked="" type="checkbox"/> Show field If checked, the <i>Page Title</i> field will appear on the node edit form for those who have permission to set the title.
Publishing options Published , Promoted to front page	
Display settings Display author and date information.	
Comment settings Open, Threading , 50 comments per page	Page Title Pattern <input type="text"/> Enter the <i>Page Title</i> pattern you want to use for this node type. For more information, please use the Page Title settings page
Menu settings	
Page Title Settings	
XML sitemap Inclusion: Excluded Priority: 0.5 (normal)	

5. Select the **XML sitemap** tab.
6. Set the **Inclusion** field to **Included**.

7. Leave the **Default priority** field set to **0.5 (normal)**.

Submission form settings Title	Changing these type settings will affect any items of this type that have either inclusion or priority set to default.
Publishing options Published , Promoted to front page	Inclusion <input type="button" value="Included"/>
Display settings Display author and date information.	
Comment settings Open, Threading , 50 comments per page	Default priority <input type="button" value="0.5 (normal)"/>
Menu settings	
Page Title Settings	
XML sitemap Inclusion: Included Priority: 0.5 (normal)	

8. Click on **Save content type**.
9. Select **Configuration** from the admin menu, then select **Clean URLs**.
10. Ensure that **Enable clean URLs** is checked, then click on **Save configuration**.
11. Select **Configuration** from the admin menu, then select **URL aliases**.
12. Select the **PATTERNS** tab.
13. In the field **Pattern for all Article paths**, enter **article/[node:title]**:

CONTENT PATHS
Default path pattern (applies to all content types with blank patterns below) <input type="text" value="content/[node:title]"/>
Pattern for all Article paths <input type="text" value="article/[node:title]"/>
Pattern for all Basic page paths <input type="text"/>
REPLACEMENT PATTERNS <input type="text"/>

14. Click on **Save configuration**.
15. Select **Configuration** from the admin menu, then select **XML sitemap**.
16. Select the **SEARCH ENGINES** tab.

17. For the field **Submit the sitemap to the following engines**, check the options **Bing** and **Google**.
18. For the field **Do not submit more often than every**, ensure that **1 day** is selected.
19. Ensure that the option **Only submit if the sitemap has been updated since the last submission** is checked:

In order to verify site ownership with the search engines listed below, it is highly recommended to download and install the [Site verification module](#).

Submit the sitemap to the following engines

Bing
 Google

Do not submit more often than every

1 day

Only submit if the sitemap has been updated since the last submission.

Custom submission URLs

Enter one URL per line. The token [sitemap] will be replaced with the URL to your sitemap. For example: <http://example.com/ping?sitemap> would become <http://example.com/ping?http://drupal7cbc12.streetfish.co.uk/sitemap.xml>.

20. Click on **Save configuration**.
21. Select the tab **REBUILD LINKS**.
22. Select **Rebuild sitemap**.
23. Select the **List tab**, and you will now see an updated tally of how many links are included in the sitemap.

How it works...

We start by editing the **Article** content type to enable the page title's **Show field** option. This adds a tab into the article's **Edit** page called **Page title settings**, which allows you to add your own custom page title that will appear in the browser's title bar. We also configure the Article's **XML sitemap** settings so that article nodes are included in the XML sitemap.

We then move on to configure the URLs for the site. We check to ensure that **Enable clean URLs** is checked. This setting is extremely important for SEO as it converts URLs on the site into a human readable format, and importantly, a search engine readable format.

We then configure the **Patterns** page in the URL aliases configuration. The **Patterns** functionality is provided by the Pathauto module, and is used to set up default naming conventions for content types, users profile pages, and taxonomies. In this recipe, we only configure the URL pattern for the Article content type, but in your site, you might want to set up patterns for all of your content types. We set the **Pattern for all Article paths** to **article/[node:title]**. This will cause all future articles to have URLs in the following form:

```
[Drupal root]/article/article-title
```



You can use the **Replacement patterns** to create a wide variety of automated URLs using the patterns settings. For example, if you created a News content type, and you wanted your news article to be submitted to Google news, then your news article URL would need to contain at least three digits. This can be automatically achieved by using the **Replacement pattern** tokens to add the node ID to the end of the URL.

After configuring the URLs, we go to the **Search engines** tab of the **XML sitemap** configuration page. We configure the sitemap to be automatically submitted to Bing and Google, no more than once a day, and only when something has changed. This step is not essential, and you can manually submit your sitemap to whichever search engines you like by using the sitemap URL listed on the XML sitemap **List** tab.

Finally, we rebuild the sitemap to ensure that it's including all of the links that we expect to see.

Securing a Drupal installation

Drupal security is a very large topic, and we would be hard pushed to attempt to cover the whole subject in one recipe! However, what we will do in this recipe is put in place some basic security checks as a security starting point.

Getting ready

To complete this recipe, install and enable the following module:

- ▶ http://drupal.org/project/security_review

You will also need to have an account on [Drupal.org](http://drupal.org) for subscribing to the Security announcements.

How to do it...

In this short recipe we will first run the Security review to identify any vulnerabilities on the site; we will then subscribe to Drupal's security announcements newsletter:

1. Select **Reports** from the admin menu, then select **Security review**.
2. Select **Run checklist**.

Untrusted roles do not have administrative or trusted Drupal permissions.	Details	Skip
Error reporting set to log only.	Details	Skip
✖ Dangerous tags were found in submitted content (fields).	Details	Skip
Drupal installation files and directories (except required) are not writable by the server.	Details	Skip
✖ Untrusted users are allowed to input dangerous HTML tags.	Details	Skip
Only safe extensions are allowed for uploaded files and images.	Details	Skip

3. Fix any items flagged in red by following the instructions on the **Details** page for each the issues.
4. When you have fixed all items, select the **Run & Review** tab.
5. Expand the **Runfieldset** and select **Run checklist**.

Untrusted roles do not have administrative or trusted Drupal permissions.	Details	Skip
Error reporting set to log only.	Details	Skip
Dangerous tags were found in submitted content (fields).	Details	Skip
Drupal installation files and directories (except required) are not writable by the server.	Details	Skip
Untrusted users are allowed to input dangerous HTML tags.	Details	Skip
Only safe extensions are allowed for uploaded files and images.	Details	Skip

6. Go to Drupal.org and log in.
7. Go to your user profile page at Drupal.org/user.
8. Select the **Edit** tab, then select the **My newsletters** secondary tab.
9. Check the option **Security announcements** and click on **Save**.

How it works...

The Security checklist module runs a number of tests on the Drupal installation to determine whether there are any security vulnerabilities from development process that have been left open and importantly, it tests that the permissions are securely configured.

We run the security review, and then check through the results to see if there are any issues. If there are, then we go to the **Details** page and follow the instructions to fix the issue. After all issues are fixed, we re-run the security review. When there are no remaining issues, all of the items on the list are green.

Finally, we log in to our [Drupal.org](http://drupal.org) account where we subscribe to the Security announcements newsletter. Once subscribed, you will periodically receive important announcements regarding security vulnerabilities in the Drupal core, or its contributed modules which you must check, and then apply the updates, if any have been made available.

There's more...

This recipe only scratches the surface of the topic of securing Drupal, and it's recommended that you spend some time doing some background research.

Drupal.org—Securing your site

Drupal has its own section dedicated to security, and it's a great place to increase your security skills:

- ▶ <http://drupal.org/security/secure-configuration>

Configuring Drupal caching

Caching is a topic at the centre of Drupal performance. It's an extremely useful feature that serves to reduce server load for anonymous users, and therefore increase page load time.

In this recipe, we will be configuring Drupal's site wide caching, caching of Views, and finally, caching of Panels.

Getting ready

In order to complete this recipe, you will need to have already created a **Panel** with some content. You will also need to have created a **View**, or enabled one of the sample views. You will also need to enable the **Page manager module**.

How to do it...

1. Select **Configuration** from the admin menu, then select **Performance**.
2. In the **CACHING** fieldset, check the options **Cache pages for anonymous users** and **Cache blocks**.
3. Set the **Minimum cache lifetime** to **5 min**.
4. Set the **Expiration of cached pages** to **1 hour**:

The screenshot shows the 'CACHING' configuration section. It includes two checked checkboxes: 'Cache pages for anonymous users' and 'Cache blocks'. Below these are two dropdown menus: 'Minimum cache lifetime' set to '5 min' and 'Expiration of cached pages' set to '1 hour'. Each dropdown has a descriptive tooltip below it.

CACHING
<input checked="" type="checkbox"/> Cache pages for anonymous users
<input checked="" type="checkbox"/> Cache blocks
Minimum cache lifetime
5 min
Cached pages will not be re-created until at least this much time has elapsed.
Expiration of cached pages
1 hour
The maximum time an external cache can use an old version of a page.

5. Click on **Save configuration**.
6. Select **Structure** from the admin menu, then select **Pages**.
7. Select **Edit** for the **Panel** you are setting the cache for.
8. Select the **Content** tab.
9. Select the cog on the content you are setting the cache for and then select **Change** under the **Caching** section.
10. In the popup, **Cache method for Existing node**, select the option **Simple cache**, then click on **Next**.
11. In the popup, **Cache settings for Existing node**, select **4 hours** for the **Lifetime** field.

12. Leave the **Granularity** set as **None**:

The screenshot shows a configuration form with two main sections: 'Lifetime' and 'Granularity'. Under 'Lifetime', a dropdown menu is set to '4 hours'. Under 'Granularity', a dropdown menu is set to 'None'. A note below the dropdown states: 'If "arguments" are selected, this content will be cached per individual argument to the entire display; if "contexts" are selected, this content will be cached per unique context in the pane or display; if "neither" there will be only one cache for this pane.' At the bottom of the form is a 'Save' button.

13. Click on **Save**.
14. Repeat steps 9-13 for all of the content on the **Panel** you wish to configure for caching, then click on **Save**.
15. Select **Structure** from the admin menu, then select **Views**.
16. Select **Edit** for the View for which you are configuring the cache.
17. Expand the **Advanced fieldset**.
18. Select the **None** link next to the **Caching** option.
19. On the **Page: Caching popup**, select the option **Time-based**.
20. Select **Apply (all displays)**.
21. For the **Query results**, and the **Rendered output** fields, leave them both set as **1 hour**, then select **Apply (this display)**.
22. On the **View edit** form, click on **Save**.

How it works...

We begin by configuring the site-wide caching options. We enable the option **Cache pages for anonymous users**. This option will store the generated HTML of a page in the database, reducing the number of database queries required to serve the page, and therefore reducing the server load. This option only applies to users who are not logged in. For logged in users, pages are re-created on each page load.

We check the option **Cache blocks**. This option causes cacheable blocks to be cached for each combination of user roles. Enabling this option will enable block caching for all users, provided that the block has been designed to allow caching.

We set the **Minimum cache lifetime** to **5 min**. This means that cached content is kept for at least 5 minutes, unless the cache is cleared manually. The time chosen for the minimum cache lifetime should be dependent on your website's needs. A low time of 1 minute is preferable for websites with lots of news content, so the cache will be refreshed after one minute. However, this can place quite a strain on a busy server, and you may need to experiment with the values to find a value that works for you.

The option **Expiration of cached pages** relates to external caching by browsers and other systems such as firewalls. We set this option to **1 hour**; however, you may find that on your website you may need to reduce this value if you are having a problem with content being unnecessarily cached.

After configuring the site-wide caching, we turn to the caching for **Panels**. First, we open an existing **Panel** and go to its **Content** tab. Then, for each item of content added to the **Panel**, we switch on **Simple caching**, and then set the **Cache settings for Existing node** to **4 hours**. The value chosen here is highly dependent on the frequency that the page is updated. If you know that the page will not be updated regularly, then it might be sensible to set a much higher value here. Conversely, if you know the content is going to be updated very frequently, then set the value below 4 hours. In this case, we haven't set the **Granularity** of the caching. The Granularity feature gives you the option to cache the Panel per individual argument supplied, or by each Context that's been configured. By leaving it set to none, only one cache will be set for the content.

The final configuration we make is to an existing **View**. We open the **View** and go to its **Caching** option. We switch on the caching for the View by selecting the **Time-based** option. We leave the **Query results** and **Rendered output** options set as **1 hour**. This means that the results of the database query can be up to an hour old; similarly, for the **Rendered output**. The database queries in Views can often be very processor intensive, and can be the cause of many bottlenecks on your site. It's quite important to consider how old you are prepared to allow the View to become before it's re-created. For a news website, you're going to want a low value for both of these options, possibly, no more than 5 minutes.

There's more...

In the preceding configuration we have seen how we can provide a significant performance boost for most websites. If you are rolling out a very large-scale site, then you will need to look at something even more powerful.

The Varnish HTTP accelerator

Varnish Cache is an application which can provide performance boosts of between 300-1000 times the previous speed. Varnish is an application which is installed on the webserver, and acts as an intermediary between the user's request and the Drupal application.

Furthermore, there is a Drupal module which provides an interface to the Varnish application's administrative functions:

- ▶ <https://www.varnish-cache.org/>
- ▶ <http://drupal.org/project/varnish>

See also

Managing site performance: <http://drupal.org/node/627252>

Running commands with the Drush tool

Drush is a fantastic time saving tool, and unless you cannot get SSH access to your server, then you should really try it out. Drush makes light work out of installing and enabling modules, and provides quick shortcuts to clearing the cache and putting the site into Maintenance mode.

In this recipe, we will be setting up and configuring Drush on your SSH server, and then running some of the most useful commands.

Getting ready

To complete this recipe, you will need to have SSH command line access to your server with the required authentication to log in and run commands. On some hosting accounts, you need to contact the host company to request that they turn on the SSH access. You will also need some knowledge of the basic UNIX commands for navigating to your Drupal root directory.

How to do it...

We will begin this recipe by installing and configuring the Drush tool. Once this is configured, we will see how to use some of the most useful functions such as installing and enabling modules, and clearing the cache.

Setting up Drush

1. Go to the Drush page on Drupal.org <http://drupal.org/project/drush> and copy the `.tar.gz` path of the latest **Recommended release** of the Drush project.
2. Log in to your webserver using your SSH client.

3. Enter `wget`, and then paste in the path to the Drush download and then press *Enter*; for example:

```
wget http://ftp.drupal.org/files/projects/drush-7.x-5.1.tar.gz
```

4. Enter `tar` followed by the name of the Drush `tar.gz` archive, then press *Enter*; for example:

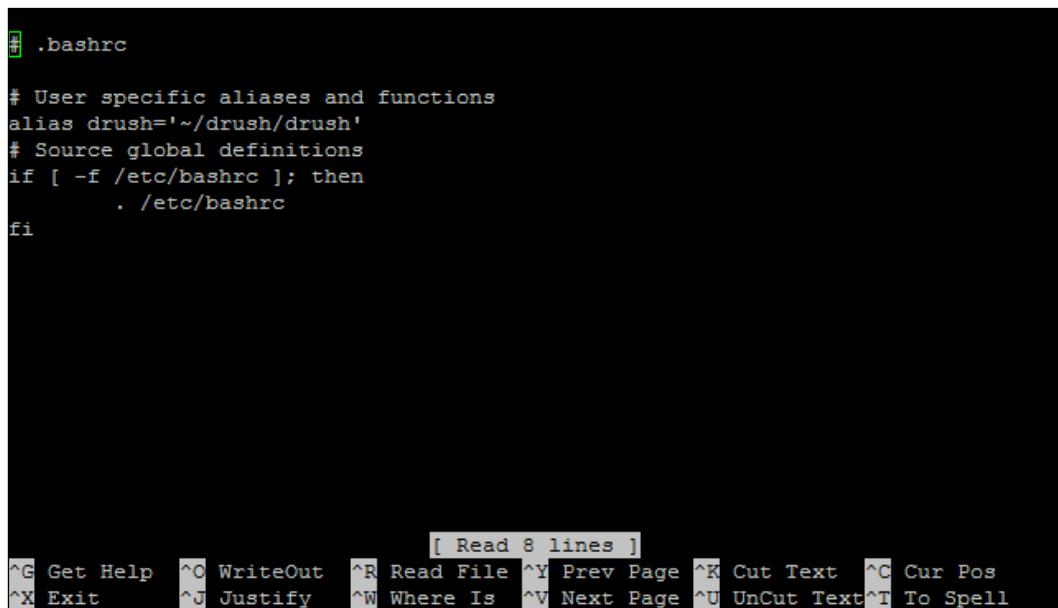
```
tar xzvf drush-7.x-5.1.tar.gz
```

5. In the root directory, open the file `.bashrc` by entering the following:

```
nano .bashrc
```

6. Under the comment `#User specific aliases and functions`, enter the following:

```
alias drush='~/drush/drush'
```



```
[ Read 8 lines ]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

7. Press *Ctrl + O* to save the file.
8. Press *Enter* to confirm the filename to save to.
9. Press *Ctrl + X* to exit the editor.
10. Restart your SSH client to enable the new alias.
11. Navigate to your Drupal installation.

12. Enter the command `drush` to test that it is working; you should receive a similar response to the following screenshot:

```
sql-sync           Copy and import source database to target database.
                  Transfers via rsync.

User commands: (user)
  user-add-role (urol)   Add a role to the specified user accounts.
  user-block (ublk)     Block the specified user(s).
  user-cancel (ucan)    Cancel a user account with the specified name.
  user-create (ucrt)   Create a user account with the specified name.
  user-information      Print information about the specified user(s).
  (uinf)
  user-login (uli)      Display a one time login link for the given user account
                        (defaults to uid 1).
  user-password (upwd)  (Re)Set the password for the user account with the
                        specified name.
  user-remove-role     Remove a role from the specified user accounts.
  (urrol)
  user-unblock (uublk)  Unblock the specified user(s).

Other commands: (make)
  make                 Turns a makefile into a working Drupal codebase.
  make-generate        Generate a makefile from the current Drupal site.
  (generate-makefile)
```

Checking the status of a site

1. Enter the following command, then press *Enter*:

```
drush status
```

2. The preceding command returns the following response:

```
Drupal version      : 7.12
Site URI          : http://default
Database driver    : mysql
Database hostname  : localhost
Database username  : [database username]
Database name      : [database name]
Database           : Connected
Drupal bootstrap   : Successful
Drupal user        : Anonymous
Default theme       : bartik
Administration theme: seven
PHP configuration  : /usr/local/lib/php.ini
Drush version      : 5.1
Drush configuration:
Drupal root         :
  directory]       : /home/[myusername]/public_html/[site
Site path          : sites/default
File directory path: sites/default/files
```

Installing modules

1. Enter the following command, then press *Enter*:

```
drush dl views ctools
```

2. The preceding command returns the following response:

```
Project views (7.x-3.3) downloaded to [success]
Project ctools (7.x-1.0) downloaded to [success]
```

3. Open your site, select **Modules** from the admin menu, and you will see that **Views** and **CTools** are now installed.

Enabling modules

1. Enter the following command, then press *Enter*:

```
drush en views
```

2. The preceding command returns the following response:

```
The following extensions will be enabled: views, ctools
Do you really want to continue? (y/n) :
```

3. Input *y*, then press *Enter*.

This returns the following response:

```
ctools was enabled successfully. [ok]
views was enabled successfully. [ok]
```

Disabling modules

1. Enter the following command, then press *Enter*:

```
drush dis views
```

2. The preceding command returns the following response:

```
Do you really want to continue? (y/n) :
```

3. Input *y*, then press *Enter*.

This returns the following response:

```
ctools was disabled successfully. [ok]
views was disabled successfully. [ok]
```

Clearing the Drupal cache

1. Enter the following command:

```
drush cc
```

2. The preceding command returns the following options:

Enter a number to choose which cache to clear.

- [0] : Cancel
- [1] : all
- [2] : drush
- [3] : theme-registry
- [4] : menu
- [5] : css-js
- [6] : block
- [7] : module-list
- [8] : theme-list
- [9] : registry

3. Input 1, then press *Enter*:

This returns the following response:

```
'all' cache was cleared in [success]
```

4. Enter the following to clear all cache in one command:

```
Drush cc all
```

How it works...

We begin by going to the Drush page on Drupal.org, where we copy the link for the latest recommended release for the Drush project. We then open the SSH client and log in.

To install Drush, we use the command `wget`, and we paste the path to the Drush package as the argument to the `wget` command. This command retrieves the Drush package, downloading it to our current working directory.

After downloading the Drush package, we unpack it using the `tar` command. We pass four arguments to the `tar` command, before the filename to unpack. These are explained as follows:

- ▶ `x`: Extract the contents of the archive
- ▶ `z`: Compress or decompress automatically
- ▶ `v`: Print verbose output; this prints the filenames as they are unpacked
- ▶ `f`: States that the archive file is given on the command line

Running the `ls` command at this point would show the unpacked Drush directory in the root. Drush is now installed, but we can currently only run it from within the `drush` directory, so we set up an alias. To do this, we edit the `.bashrc` file (using the Nano editor) in the root of the server and add the following line:

```
alias drush='~/drush/drush'
```

This command will means that the system can know to look in the `/drush/drush/` folder any time the `drush` command is entered. We could have just entered the `alias` command directly into the command line, but the alias would only persist as long as the session. To enable the new alias, we restart the SSH client.

After restarting the SSH client, we navigate to the Drupal installation directory on the server, and enter the `Drush` command to test that it is working.

The first Drush command that we use is the `status` command; this simply returns some top-level information about the system such as its database name, Drupal version and the default theme.

The next Drush command that we use is the `d1` command, this is the download command. This will download the modules named in the arguments and place them into the site's modules directory. All we need to do is enter `drush d1`, followed by a list of modules to install.

The next Drush command that we use is `drush en`, which is the enabling command. We simply enter the `en` command, followed by the name of the module to enable. If the module has dependencies, then you will be prompted to enable them.

After enabling the **Views** and **CTools** modules, we disable them using the `drush dis` command, followed by the module names.

Finally, we see how to use the very useful clear cache command. If we enter `drush cc`, we see a list of the available options which prompt us to decide which parts of the cache to clear. We choose to clear all. We can do this in one command by adding the name of the option after the arguments, for example:

```
drush cc all
```

There's more...

Hopefully, we are now beginning to see the benefits that using Drush can bring. Drush is sure to increase your efficiency as a developer, but you can become even more efficient.

Drush make

Drush make is a feature of Drush that allows you to create flat file scripts, which can download a Drupal installation and the specified modules to your server.

Installing Drush using PEAR

There is an easier way to install Drush on your server, by using the PEAR installer. Type the following commands into your command line interface:

```
pear channel-discover pear.drush.org  
pear install drush/drush
```

See the Drush project page for more details.

See also

- ▶ The Drush project page on the Drupal site: <http://drupal.org/project/drush>
- ▶ Drush – Drupal shell utility: <http://drupal.org/node/477684>
- ▶ Drush make: <http://drupal.org/node/625094>
- ▶ The Drush website: <http://drush.ws/>

Index

A

adaptive themes 221

Add this

- about 187
- setting up 187-189
- working 189

Administrators 180

advanced content type

- creating 79-84
- working 84, 85

Aggregator module

- about 180
- used, for displaying Twitter feed 180

App display name 175

App domain field 175

App namespace 175

archived content block and view

- creating 113-117

arguments

- passing to view, from Panel 104

attachments

- used, for extending views output 124-128

B

backup system, Drupal

- alternative backup destinations, using 284
- manual backups, creating 283
- restoring, from backup file 283
- setting up 274-280
- working 281, 282

basic content type

- creating 68-70
- menu system, creating 71
- working 70

basic page

- adding, to main menu 23, 24
- creating 22, 23
- working 24, 25

biographies page

- building, Views used 257

biography field

- adding, to user profile 253-256

block region

- adding, to theme 55, 57

blocks

- about 41
- adding, to page 134-136
- conditional display 64
- creating, Views module used 52-54

block types 44

block visibility configuration

- about 64, 65
- by content type 65
- by URL 65

Boost module 224

C

caching

- about 290
- configuring 221, 223

Castles 191

Chaos tools

- URL 85

CKEditor

- configuring 26
- downloading 26

CKEditor 3.6.2 26

commands

- running, Drush used 294-296

comment fields
expanding 38

comments
configuring 36, 37
permissions 38

Commerce Kickstart distribution
downloading 13
URL 13

complex views
building, relationships used 118-122

conditional display, block
about 64
block visibility, setting with modules 65
visibility by content type 65
visibility by URL 65

Contact translation feature 241

content
editing 29, 30
image, adding 31

content block
creating 42, 43
working 43

content nodes
tabbed settings, adjusting 33, 34

content type
documents, linking to 161-164
video, adding 165-167

content types
advanced content type, creating 79-84
basic content type, creating 68-70
custom content importer, building 85-88
field types, installing 75, 77
forum, building 89, 90
image format, applying 73, 74
output, configuring 71-73
PayPal integration, adding 181-185

content type translation
enabling 232-235

contributed modules 189

cron 17

Crop effect 75

custom content importer
building 85-88
working 88, 89

custom language
adding 229

custom page layout
creating, Layout builder used 143-147
saving, for reuse 147
working 147

custom text
adding, to page 130-133

D

Date module
features 80
URL 80

Desaturate effect 75

Devel module 246

device-specific theme
using 217

directory
using, for mobile site 217

document categorization 161

document content type
creating 155, 156
working 157

Document nodes 160

documents
linking, to content type 161-164

Drupal
about 8
Add this social bookmarking service,
setting up 187
backup system, setting up 274, 275
basic page, adding to main menu 23, 24
basic page, creating 22
biography field, adding to user
profile 253,-256
blocks 41
block types 44
caching, configuring 221, 223, 290-292
comments, configuring 36
content types 68
content type translation, enabling 232-235
downloading 8
Drush tool 294
existing content, editing 29-31
Google Map, adding to content 190
HTML output, overriding of content
type 202-211
image blocks 44

installing 8-10
integrating, with Facebook 172
language switching block, displaying for end users 235, 236
mobile theme, installing 218-220
modules, installing 14
multilingual view, creating 237-240
PayPal integration, adding to content types 181
prerequisites 8
profile pictures grid view, building 258-268
regions of theme, viewing 44
RSS feeds, publishing 39
security 288
SEO 284
site search, setting up 16, 17
submenu block, creating 45-47
Superfish menu block, creating 49
system maintenance 270
tablet theme, installing 218-220
theme compression, configuring 221, 223
theme, creating from scratch 211-214
themes, installing 14
Twitter feed, displaying 176
user accounts, creating 244, 246
user notification, setting up 249-251
user roles, managing 246-248
working 11
WYSIWYG editor, installing 26, 27

Drupal 7 release
downloading 8

Drupal Backup and migrate module
URL 284

Drupal cache
clearing 297

Drupal caching
configuring 290-292
working 292, 293

Drupal distributions
Commerce Kickstart distribution 13
installing 13, 14
requisites 13

Drupal installation 12
auto-installers 11
database table prefixes 11
different languages 12
installing on Windows environment 12
requirements page, verifying 12
securing 288, 289
uploading, through CPanel 11

Drupal security
about 288
working 290

Drupal site
securing 290

Drupal theme
about 55
block region, adding 55-57
Superfish menu block, adding 59

Drupal Twitter Feed
displaying 178

Drush
about 294
configuring, on SSH server 295
Drupal cache, clearing 297
installing, PEAR used 300
modules, disabling 297
modules, enabling 297
modules, installing 297
reference links 300
setting up 294
site status, checking 296
working 298, 299

Drush command-line tool used
used, for installing modules 16

Drush make 300

dynamic view
adding, to page 137, 139
creating 98-103
working 103, 139

E

Editor role 246

EPSA Crop module 75

URL 75

existing node

adding, to page 134

F

Facebook

integrating, with Drupal 172-175

Facebook application

creating 172

Facebook-Druapl integration

- about 172
- App display name 175
- App domain field 175
- App namespace 175
- Facebook account, activating 172
- Facebook application, creating 172, 173
- Facebook login module, configuring 172, 173
- Site URL field 175
- working 175

Facebook OAuth module

- configuring 172, 173
- using 172

favicons

- implementing 215

Feeds module

- URL 85

Field collection module

- about 79
- URL 79

field_slideshow module 150**field types, contenttype**

- installing 75-77
- working 78

forum

- accessing 91
- building 89, 90
- working 91

Forum access module

- URL 91

forum content type

- extending 91

Forum translation feature 241**full e-commerce online shop**

- creating 186

G**generate users 246****GMap module**

- about 191
- configuring 194

Google Map

- about 190
- adding, to content 190-194
- working 194

granularity 36**grid view, profile pictures**

- building 258-267

H**HTML output**

- overriding, of content type 202-211

hyphens 172**I****image**

- adding, to existing content 29-31
- alternative ways, of adding to content 32
- properties 32
- size 32

image blocks

- image blocksabout 44

image format

- applying, to content type 73, 74

image format, content type

- Crop effect 75
- Desaturate effect 75
- manual cropping 75
- Rotate effect 75

ImageMagick Raw Effect module

- about 75
- URL 75

image properties 32**image size 32****Indexing status category 18****installation**

- Drupal 8

interface translation

- managing, with Locale module 230, 232

Internationalization module 241**J****jcarousel module 150****Job scheduler module**

- URL 85

jquery.cycle 151

L

language

installing, Locale module used 226-229

language detection methods 229

language icons 237

language switcher drop down 237

language switching block

displaying, for end users 235, 236

latest news block

creating 104-106

working 106, 107

Layout builder

used, for creating custom page

layout 143-147

libraries module 150

Link module

about 14

installing 14

working 15

Locale module

about 225, 226

interface translation, managing 230, 232

language, installing 226-229

Location field 191

Location module

about 191

configuring 194

M

manual backups

creating 283

markItUp 28

Media module 169

media_youtube module 166

mega-footer menu

about 59

blocks, adding 63

creating 59-62

items, adding 63

regions, adding 63

working 62

Menu view

creating 71

mobile theme

installing 218-220

Mobile tools module

about 215

using 216, 217

modules

installing 14

installing, Drush command-line tool used 16

manual installation 16

multilingual view

about 237

creating 237-240

multi-site Drupal installation

creating 18-20

working 20

N

news image grid view

creating 107-110

working 110

news listing view

alternative listings layouts 98

creating 94-97

working 97

nodes 33

notification system

building 253

O

output, content type

configuring 71-73

P

page

block, adding 134-136

custom text, adding 130-133

dynamic view, adding 137, 139

existing node, adding 134

visibility, configuring 140, 142

Page manager module 290

Panels module

about 98, 129

features 130

Pathauto module 36

Paypal button code method

expanding 186

PayPal integration
adding, to content types 181-185
full e-commerce online shop, creating 186
working 185, 186

PEAR
used, for installing Drush 300

Profile 2 module 257

profile pictures
grid view, building 258-267

R

randomly selected list of images
creating 111, 113

regions of theme

viewing 44

relationships

creating, reference modules used 122

Rotate effect 75

RSS feed

about 39

publishing 39

working 40

S

Sasson theme

about 220

support, for Google fonts 221

support, for SASS 220

Security checklist module 290

security, Drupal. *See* **Drupal security**

SEO

about 284

bespoke URLs, configuring 285-287

custom page titles, configuring 285-287

optimizing methods 284

XML sitemap, generating 285-287

Session detection method 229

settings, WYSIWYG

Apply source formatting 28

Cleanup and Output section 28

CSS options field 29

Editor CSS option 29

Force clean up on standard paste 29

Paste from Word function 29

Verify HTML 28

simple document library
categorization, adding 161
creating 158, 159
working 160, 161

simple slideshow carousel

creating 150-153
working 154

site search

setting up 16, 17
working 17, 18

Site URL field 175

sub-domain

using, for mobile site 217

submenu block

creating 45-47
fixed starting item 48
Maximum depth field 48
working 47, 48

subtheme, Bartik theme

creating 55-57

Superfish menu block

adding, to theme 59
creating 49, 50
main navigation, replacing with 52
working 51

Superfish module

about 49

URL 49

system maintenance

about 270

Drupal modules, updating 271-274

maintenance mode 274

working 274

T

tabbed settings, for content nodes

adjusting 33, 34

working 34

tablet theme

installing 218-220

template integration 189

text search filter

adding, to view 122, 123

theme

creating 198-201

creating, from scratch 211-214

theme compression
configuring 221, 223
theme display breakpoints 220
theme inheritance 201
theme override functions 202
themes
installing 14
theme screenshot
about 215
adding 215
third-party Twitter modules
URL 181
TinyMCE 28
Trigger module 249
Triggers module 38
Twitter feed
displaying 176-180
displaying, Aggregator module used 180
Twitter Profile Widget 181
Twitter widget
using 176

U

underscores 172
user accounts
creating 244, 246
creating automatically 246
creating, from CSV 246
User detection method 229
user-mini Image format 121
user notification
setting up 249-253
user profile
biography field, adding 253-257
user roles
managing 246-249

V

Varnish Cache 224, 293
Varnish HTTP accelerator 293
Video content node 166
Video content type
creating 165-168
videos, embedding from other resources 169
videos, embedding in WYSIWYG 169
working 168
views
adding, Panels used 98
Views module 40
features 94
used, for creating block 52-54
visibility, of page
configuring 140, 142

W

WebMatrix software 12
Website Payments Standard 181
WYSIWYG editor
about 21, 149
configuring 26, 27
installing 26
markItUp 28
settings 28
TinyMCE 28
used, for text formats 28
working 28
YUI Editor 28

Y

YUI Editor 28

Z

Zen
theme, creating 198-201



Thank you for buying Drupal 7 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.

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. 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.

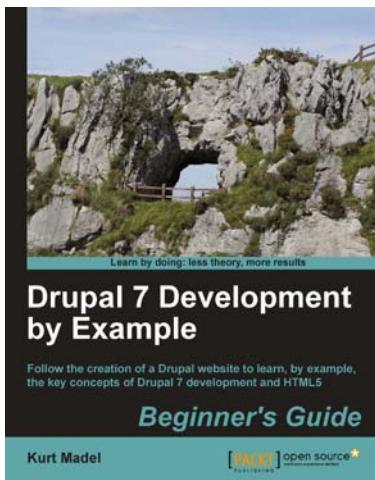


Drupal 7 Multilingual Sites

ISBN: 978-1-849518-18-5 Paperback: 140 pages

A hands-on, practical guide for configuring your Drupal 7 website to handle all languages for your site users

1. Prepare your Drupal site to handle content in different languages easily
2. Apply the numerous multilingual modules to your Drupal site and configure it for any number of different languages
3. Organize the multilingual pieces into logical areas for easier handling



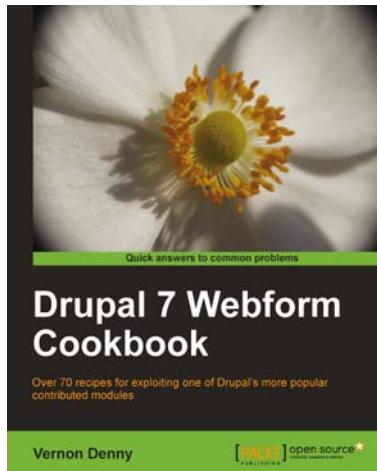
Drupal 7 Development by Example Beginner's Guide

ISBN: 978-1-849516-80-8 Paperback: 366 pages

Follow the creation of a Drupal website to learn, by example, the key concepts of Drupal 7 development and HTML5

1. A hands-on, example-driven guide to programming Drupal websites
2. Discover a number of new features for Drupal 7 through practical and interesting examples while building a fully functional recipe sharing website
3. Learn about web content management, multi-media integration, and e-commerce in Drupal 7

Please check www.PacktPub.com for information on our titles



Drupal 7 Webform Cookbook

ISBN: 978-1-849516-48-8 Paperback: 274 pages

Over 70 recipes for exploiting one of Drupal's more popular contributed modules

1. Build feature-rich Webforms that inspire awe and amazement from the comfort of your web browser
2. Invoke your creative engine with stunning modules that extend an already comprehensive module – even create your own extensions
3. Explore every known nook and cranny of Webform in detailed byte-sized steps, including handling common pitfalls and caveat



Drupal 7 Views Cookbook

ISBN: 978-1-849514-34-7 Paperback: 218 pages

Over 50 recipes to master the creation of views using the Drupal Views 3 module

1. Brand new recipe examples using the all new Views 3 UI
2. A wide variety, including multi-display and programmatic views
3. Easy-to-follow recipes with plenty of screenshots and demonstrations

Please check www.PacktPub.com for information on our titles