

Earl Miles  
Lynette Miles  
With Emma Jane Hogbin  
and Karen Stevenson



Foreword by Dries Buytaert,  
founder and project lead of Drupal,  
CTO of Acquia

# Drupal's Building Blocks

Quickly Building Web Sites  
with CCK, Views, and Panels

**Developer's Library**



Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales  
(800) 382-3419  
corpsales@pearsontechgroup.com

For sales outside the United States, please contact:

International Sales  
international@pearson.com

Visit us on the Web: [informit.com/aw](http://informit.com/aw)

*Library of Congress Cataloging-in-Publication Data*

Drupal's building blocks : quickly building web sites with cck, views,  
and panels / Earl Miles ... [et al.].  
p. cm.

Includes bibliographical references and index.

ISBN 978-0-321-59131-9 (pbk. : alk. paper)

1. Drupal (Computer file) 2. Web sites—Authoring programs. 3. Web  
site development. I. Miles, Earl.  
TK5105.8885.D78D77 2010  
006.7'8—dc22

2010043527

Copyright © 2011 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc.  
Rights and Contracts Department  
501 Boylston Street, Suite 900  
Boston, MA 02116  
Fax: (617) 671-3447

ISBN-13: 978-0-321-59131-9

ISBN-10: 0-321-59131-3

Text printed in the United States on recycled paper at RR Donnelley in  
Crawfordsville, Indiana.

First printing, December 2010

**Associate  
Publisher**  
Mark L. Taub

**Executive Editor**  
Debra Williams  
Cauley

**Development  
Editor**  
Michael Thurston

**Managing Editor**  
John Fuller

**Full-Service  
Production  
Manager**  
Julie B. Nahil

**Project  
Management**  
LaurelTech

**Copy Editor**  
Jill E. Hobbs

**Indexer**  
Jack Lewis

**Proofreader**  
Charles  
Roumeliotis

**Technical Reviewers**  
Jen Lindner  
Andy Wilson  
Chris Hanson  
Clay Robeson

**Publishing  
Coordinator**  
Kim Boedigheimer

**Interior and Cover  
Designer**  
Gary Adair

**Compositor**  
LaurelTech

*This page intentionally left blank*

# Contents at a Glance

Contents	<b>ix</b>
Foreword	<b>xvii</b>
Preface	<b>xix</b>
Acknowledgments	<b>xxiii</b>
About the Authors	<b>xxiv</b>

## **I: Content Construction Kit**

<b>1</b>	Introducing CCK and Nodes	<b>3</b>
<b>2</b>	Field Concepts	<b>17</b>
<b>3</b>	Deeper into Fields	<b>39</b>
<b>4</b>	Themes and CCK	<b>59</b>
<b>5</b>	CCK API	<b>75</b>

## **II: Views**

<b>6</b>	Relational Databases	<b>101</b>
<b>7</b>	Creating Views	<b>109</b>
<b>8</b>	Arguments, Exposed Filters, and Relationships	<b>137</b>
<b>9</b>	Theming Views	<b>153</b>
<b>10</b>	Query Optimization	<b>177</b>
<b>11</b>	Views API	<b>185</b>

## **III: Panels**

<b>12</b>	Introducing Panels	<b>217</b>
<b>13</b>	Creating Panels	<b>225</b>
<b>14</b>	Context, Relationships, and Arguments in Panels	<b>257</b>
<b>15</b>	Panels Theming	<b>275</b>
<b>16</b>	Site Deployment	<b>291</b>

**IV: Appendices****A** Other Useful Modules **303****B** Reporting Issues **309****C** Views API Handlers and Plugins **315****Index 327**

# Contents

Foreword	xvii
Preface	xix
Acknowledgments	xxiii
About the Authors	xxiv

## I: Content Construction Kit

### 1 Introducing CCK and Nodes 3

The Node System	3
Why Nodes Are Important	4
Default Content Types	4
Parts of a Node	5
Why Add Fields to Nodes?	7
Quest for the Grail: How CCK Was Born	8
Getting Started with CCK	10
Creating a New Content Type	11
Summary	15

### 2 Field Concepts 17

What You Should Know Before Creating Fields and Content Types	17
The Content Type Pages	17
Identification	18
Submission Form Settings	19
Workflow Settings	21
Comment Settings	22
Fields, Export, and Import	22
Creating New Fields for Content	24
Adding Fields	24
Data Types	26
Widgets	29
Putting the Parts Together	30
Changing the Field Display	33
Sharing Fields	36
Summary	38

<b>3</b>	<b>Deeper into Fields</b>	<b>39</b>
	Choosing Field Types	39
	Using Text	40
	Using Numeric Types	40
	Using Node Reference	41
	User Reference	42
	Constraining Data with Widgets	43
	Date Module	48
	Computed Fields	51
	Link and Email Fields	54
	Highly Visual Media	55
	FileField	55
	ImageField	56
	ImageAPI, ImageCache, and ImageCache UI	56
	Summary	57
<b>4</b>	<b>Themes and CCK</b>	<b>59</b>
	Theme Basics	59
	CCK Specific Theming	62
	Formatters	62
	Field Templates	62
	Node Templates	65
	Excluding Fields	68
	Node Reference	68
	Helper Modules	69
	Theme Developer	70
	Contemplate	71
	Summary	73
<b>5</b>	<b>CCK API</b>	<b>75</b>
	Using the CCK API	75
	The Field Model	76
	Creating New Field Types, Widget Types, and Formatters	76
	Creating Custom Field Types, Widget Types, or Formatters	77
	Field Type Modules	79

Widget Type Modules	<b>85</b>
Formatter Modules	<b>87</b>
Creating Field Instances Using Content Copy	<b>91</b>
Creating Field Instances with the CRUD API	<b>91</b>
Creating Data for CCK Fields	<b>95</b>
Miscellaneous Helper Functions	<b>97</b>
Summary	<b>98</b>

## **II: Views**

<b>6 Relational Databases</b>	<b>101</b>
Drupal, SQL, and the Emergence of Views	<b>101</b>
Pronouncing SQL	<b>101</b>
The Drupal “Learning Cliff”	<b>102</b>
The Basics of Relational Databases	<b>102</b>
Rows and Fields	<b>103</b>
Keys	<b>103</b>
Filtering and Sorting	<b>105</b>
Filtering	<b>105</b>
Sorting	<b>106</b>
Joins	<b>106</b>
From SQL to Views to Human Language	<b>107</b>
Summary	<b>108</b>
<b>7 Creating Views</b>	<b>109</b>
Views UI	<b>109</b>
List	<b>110</b>
Add	<b>112</b>
Import	<b>125</b>
Tools	<b>125</b>
Showing Your Views to the World: Creating Displays	<b>131</b>
Blocks	<b>132</b>
Pages	<b>133</b>
Attachments	<b>135</b>
Feed	<b>135</b>
Summary	<b>136</b>



<b>8</b>	<b>Arguments, Exposed Filters, and Relationships</b>	<b>137</b>
	Arguments	137
	Arguments as Filters	138
	Configuring an Argument	139
	Using Arguments as Part of a View	142
	Exposed Filters	145
	Relationships	148
	Summary	151
<b>9</b>	<b>Theming Views</b>	<b>153</b>
	An Overview	153
	Classes in Views	153
	Template Files	156
	The Display Templates	159
	View Styles	160
	The Row Templates	161
	Other Templates	162
	Working with Templates	164
	Rescan the Template Files	165
	Debugging	167
	Printing Default Messages for Empty Fields	167
	Grouping in a Template	169
	Summary	175
<b>10</b>	<b>Query Optimization</b>	<b>177</b>
	Balancing Development Time Against CPU Time	177
	Sticking with What Views Gives You	177
	When You Need More Than Views	178
	Determining Query Performance	178
	Embedding Queries	179
	EXPLAIN	180
	Indexing Versus Caching	182
	Experimenting with Your Site	183
	Summary	183

**11 Views API 185**Data Architecture **185**Object-Oriented Programming **186**Base Tables and Relationships **191**The Objects Involved in a View **192**The Views API **199**The Life Cycle of a View **201**View Execution Cycle **201**Executing a Views Display **203**Execution-Related Hooks **204**The Database Schema and Data Hook **206**Relating Tables to Each Other **206**Declaring Tables in `hook_views_data()` **207**Declaring Fields on Tables **209**Handlers Versus Plugins **210**Handlers **210**Plugins **212**Summary **213****III: Panels****12 Introducing Panels 217**Introduction to Panels **217**A Brief History of Panels **217**Push and Pull: How Panels Is Different **218**Point-and-Click Layout **219**Context **221**Pluggable Architecture **221**Modules **222**Panels Package **222**Chaos Tool Suite **223**Summary **224****13 Creating Panels 225**Your First Panel **225**The Panels Dashboard **226**Panel Pages **227**

Panel Nodes	233
Mini-Panels	236
Adding Content to Panels	237
Adding Content Panes	238
Caching	242
Configuring Existing Content Panes	243
Access Rules	244
Overriding Core Display Pages	246
Node View	248
Taxonomy	251
User View	252
Overriding Core Node Editing Pages	253
Summary	255
<b>14 Contexts, Relationships, and Arguments in Panels</b>	<b>257</b>
Contexts	257
Contexts in Panel Pages	258
Taxonomy	262
Arguments in Panes	264
The Add Content Modal for Views	266
View Pane Displays	268
Relationships	271
Using Relationships	271
User Reference and Node Reference	272
Summary	274
<b>15 Panels Theming</b>	<b>275</b>
Layout	275
Flexible Layout	275
Changing Layouts	279
Stylizer	280
Working with Styles	282
CSS in the Panels UI	284
CSS in Source Code	285
Identifying a Particular Pane	286
Other Stylistic Changes	289
Summary	290

**16 Site Deployment 291**Configuring Your Development Environment **291**Content Versus Structure **291**Source Control **292**Moving to Production **293**Keeping Development Separate from Production **293**Testing Your Changes **293**Documenting Your Work **294**Exporting Your Structures **294**Exporting CCK **295**Exporting Views **296**Exporting Panels **298**Helper Modules **299**Deploy **299**Features **299**Drush **299**Summary **300****IV: Appendices****A Other Useful Modules 303**Extending the Use of Your Modules **303**General Modules **303**Pathauto **303**Views **304**Views\_or **304**Nodequeue **304**Flag **305**Views Slideshow **305**Views Bonus Pack **305**Views Attach **306**Views Import **306**ApacheSolr Views **306**SimpleViews **307**Views Bulk Operations **307**Views Datasource **307**

Sheetnode	<b>307</b>
CCK	<b>307</b>
Calendar	<b>308</b>
Panels	<b>308</b>
Advanced Profile Kit	<b>308</b>
Total Control Admin Dashboard	<b>308</b>
<b>B Reporting Issues</b>	<b>309</b>
Submit a Complete Report	<b>309</b>
Read the Documentation	<b>310</b>
Check Other Sources	<b>311</b>
Know the Difference between a Bug and a Support Request	<b>312</b>
Stay on Topic	<b>312</b>
Understand the Life Cycle of a Bug	<b>312</b>
Be Patient	<b>313</b>
Remember That You're Asking for Someone Else's Time	<b>314</b>
Contribute Back	<b>314</b>
<b>C Views API Handlers and Plugins</b>	<b>315</b>
Views Handlers	<b>315</b>
Field Handlers	<b>315</b>
Sort Handlers	<b>316</b>
Filter Handlers	<b>316</b>
Handlers for Arguments	<b>316</b>
Relationship Handlers	<b>317</b>
Views Plugins	<b>317</b>
Display Plugins	<b>317</b>
Style Plugins	<b>317</b>
Row Plugins	<b>318</b>
Views Classes	<b>318</b>
<b>Index</b>	<b>327</b>

## Foreword

There was a time, in the 1950s, when to be a computer programmer you had to be something of an electrical engineer. You had to be handy with wire cutters and strippers and be willing to get your hands dirty—literally. That all changed over the decades, and programming a computer became a simple feat by contrast. Still, it remained the domain of only a few people with the proper education and technological sense. It was the advent of microcomputers and the Internet that made the world of technology more accessible, or at least began the process of attracting more people.

It was that time and those elements—that first major wave of public inclusion—that called for easier methods and for better tools for programming, for making use of computers, and for communicating information throughout the world.

It seems that each decade—perhaps not exactly in 10-year increments—brings with it a new wave of technology that makes the use and manipulation of technology accessible to more people. Each period begins with only people of certain technology prowess being able to participate fully. But in time the demand becomes so great, and the desires of the greater community so intense, that new innovations are achieved and new ways are determined in which more people can be part of the creation process and not just be on the receiving end.

It was just 15 years ago or so that Web design required an in-depth understanding of HTML, skills in network configuration, and the ability to program using less-than-intuitive programming languages to be able to do more than create a few flat Web pages. To be able to create forms, allow users to enter information themselves on a site, and provide many of the features that are commonplace today required the advanced and diverse programming skills of a Web developer—not to mention a sense of design, an understanding of marketing, and good writing abilities. So, many sites were either poorly constructed or else were the result of a heavily orchestrated organization that employed many people from diverse backgrounds. Web design was simply inaccessible for most people and organizations.

That has all changed as well and is continuing to change. We're in the middle of a new period of accessible technology, it seems. Drupal is changing the way Web sites are built. While Drupal can be used as a Web programming framework, it doesn't have to be. Unlike many other Web design tools, you don't have to be a programmer to build a Web site with Drupal.

In the Drupal world, many people build Web sites very easily. You just decide what you want on your Web site—text, photographs, a blog, places for visitors to comment, a feed from your Twitter account, and many, many other things—and then download the modules you need based on your wish list (you can have all that you wish for now), install each module, do a bit of configuring through your Web browser (mostly pointing and clicking, with the occasional typing of content), and you're done. Zero programming is required. It's that easy.

The Drupal community has created thousands of modules, all freely available from the drupal.org Web site. It's a credit to the collective efforts of thousands of smart people working together for years, not only for their own interests, but even more so for the benefit of others. Each module alters and extends Drupal's core capabilities and adds new functionality to a Drupal site. Owing to the vast amount of modules available from the Drupal community, the number of distinctly different sites that can be built using Drupal is unlimited, and the number that have already been built using Drupal is extensive. The speed at which sites can be assembled using Drupal and Drupal modules is surprising and unmatched. Not a single proprietary content management system has the depth and breadth of Drupal—not to mention that it's free.

Nevertheless, two contributed modules stand out from the rest: Content Construction Kit (CCK) and Views. Not only are they the most popular modules, but they are also two of the most flexible modules. I have repeatedly been surprised by how Web developers use Drupal, and what they build using CCK and Views. More than once, I've been shown "new tricks" of what can be done with CCK and Views without a single line of programming. The world of CCK and Views is an interesting one. The true depth and richness of these two modules have been mastered by only a few people, because ultimately the limits of what you can do with these modules has more to do with the data provided to them than it has to do with the capabilities of the site builder. The Panels module, while not standing out quite as strongly as Views and CCK, allows site builders the opportunity to tune their sites more carefully to look and feel the way they want. It gives them large amounts of control and organization, again without needing to write a single line of code. It, too, has allowed surprising systems with only a few tricks.

Behind the code that makes these modules work is a strong community of committed volunteers. One of these people is Earl Miles, a coauthor of this book. As an active member of the Drupal community for many years now, he has contributed a great deal to the direction of Drupal. More specifically to the topic of this book, he is a key contributor to CCK and the principal author of Views and Panels. I cannot think of a better person to write about these particular modules. Coauthor Lynette Miles, in contrast, is not a developer at all, and yet these modules have allowed her to contribute to the Drupal project by providing support for the usage of these modules both on drupal.org and in IRC. Her knowledge of the questions people ask when learning to use these modules is instrumental in understanding how to explain these sometimes difficult concepts to users.

Even if you're already a seasoned user of CCK, Views, or Panels, I have no doubt that this book will provide you with several new techniques and methods for getting the most out of these extremely essential modules. It certainly has for me.

*Dries Buytaert*  
*Founder and project lead of Drupal,*  
*CTO of Acquia*  
*October 2010*

## Preface

Drupal is an open source software package that is offered for free to download, modify, and use. It has been implemented by thousands of people around the world and is used by millions of people daily as the basis for discussion Web sites, community portals, corporate intranets, e-commerce Web sites, vanity Web sites, resource directories, image galleries, podcasts, and more. By choosing to use Drupal, you are accessing not only an award-winning Web platform, but also its vibrant community.

Often referred to as “The Big Three,” the Content Construction Kit (CCK), Views, and Panels modules have fundamentally changed the way developers, site builders, and designers create Drupal Web sites—and yet they are all contributed modules. In this book, the core contributors to these three suites of modules teach you how to build better Web sites. The modules described are widely considered essential modules that will be installed on almost every site. They allow for a level of customization that is unparalleled in the market, and are a key reason that Drupal is being chosen over its competition.

The book assumes you are familiar with how to install Drupal and enable modules. Web developers and administrators of Drupal Web sites are the target audience, although the book is written so that devoted Drupal enthusiasts can fully customize their sites using the information provided here.

## Part I—Content Construction Kit

Content Construction Kit is a module that allows you to define the data that makes up your site’s content types. It lets you add new fields chosen from a variety of field types, such as text, numbers, dates, and even references to other content. It handles input forms and provides a variety of output styles for each field. Throughout the first part of this book, you will learn how to use CCK to customize your data objects to conform to your needs, rather than making your needs conform to the core content types.

### Chapter 1

In Chapter 1, we explain the basic concepts needed to understand the powerful but complicated creature known as CCK, including how it came to be, how the basic Drupal structure is defined, and why the level of flexibility and customization offered by the node system is important.

### Chapter 2

Expanding on the general usage of content types and fields, Chapter 2 delves into how CCK works its magic, both from an administrative user interface (UI) point of view and within the Drupal database itself. To do so, we explore two potential Web sites: a homebrewer’s journal and a T-shirt sales site.



## Chapter 3

There are an extensive number of ways you can use content fields to create your Web site. Understanding and using fields and helpers for those fields creates possibilities for any kind of data. In Chapter 3, we dig more into field types and consider why you might want to use one type of field over another in your content type. We also look at some field types that you might want to add, but that are not part of the core CCK package.

## Chapter 4

Now that your content is created, it's time to make it look professional and easy to read. CCK does a great job of allowing you to add plenty of customized content. What it doesn't do as well is display the data in a fashion that is clean and nicely readable for users. In Chapter 4, we take a look at the theme system and how CCK interacts with it.

## Chapter 5

CCK includes methods that PHP developers can use to create fields outside of the user interface. This creates even more flexibility, but requires a definite knowledge of the PHP language as well as familiarity with Drupal's development style. In this chapter, we delve into integrating CCK with other modules.

## Part II—Views

The Views module is a powerful query builder designed to simplify the task of building custom query displays. It accomplishes this feat by providing lists of all table and field information that it knows of and letting the user assemble items from these lists together. After a complete rewrite for Drupal 6, Views has a new interface with more options than ever before. With the addition of a live preview and query display, site builders can nail down their displays in a way that was previously impossible to do without making changes that can affect all users.

## Chapter 6

Drupal relies on an SQL database to store information, and it currently supports MySQL and PostgreSQL. Properly using Views requires an understanding of how the database stores data, how it is related across various tables, and how Drupal works with the database to retrieve data. This chapter is directed toward newer users and programmers.

## Chapter 7

In Chapter 7, we focus on the Views UI, including how each function works. We look at how each piece creates part of a query, and how the results of those queries fit into pages and blocks. We also discuss the most important filters you may need as well as how to create relationships between node content that does not otherwise share information.

RSS, styles and fields, and the Views Bonus Pack are other important topics when determining what you want out of your view; they are also covered in this chapter.

## **Chapter 8**

Supplying arguments to Views is one of the ways the Views module becomes even more powerful and flexible. Relationships bring data together in new ways, and expand the information available to the rest of Views. Chapter 8 describes how to customize your views even more through the power of relationships, arguments, and filters.

## **Chapter 9**

Views can be themed just like anything else in Drupal. The Views module provides an entirely new level of classes, theming templates, and strategies over its predecessor. In Chapter 9, we discuss the template files and their contents, change some CSS, and look at how we can approach rendering data by multiple methods.

## **Chapter 10**

One of the biggest questions facing the developers of any software installation of any kind is, “How much time will each part of this application take?” Entire software packages exist to measure this kind of information. For some people, optimization is the key to a well-run and well-maintained site. For others, this issue represents a giant hassle. In Chapter 10, we provide a few suggestions as to when and why it might be appropriate to do some customization to your Views-generated queries.

## **Chapter 11**

In Chapter 11, we explore the nuts and bolts of how Views is put together—at the code level. You will learn about the data architecture of Views, the life cycle of a view, and its database schema. We also introduce the plugins and handlers needed to control custom queries and formatted output.

## **Part III—Panels**

Now it’s time to really customize how you want your site to look. The Panels module supplies a group of standard layout templates. In this part of the book, you learn how to create panels that override default page layouts, explore how to theme these layouts, and get an introduction to the Panels API.

## **Chapter 12**

The core functionality of the Panels module is layout; designing the layout is when things start to look polished. With an understanding of Panels, administrators can create a style that is easily applied to every page of a site, or a different style for every page. Chapter 12 provides an introduction to how Panels works.

## **Chapter 13**

In Chapter 13, we investigate the Panels UI, including how each part fits together to create a wide range of panels. You learn how to create your very first panel, add content to a range of panel types, and override the core display pages for each of your site's content types.

## **Chapter 14**

Panels incorporates a few major features that can take you from the basics of Web site development to real complexity. Using arguments, relationships, and contexts, you can build connections between pieces of content in your panel layouts.

## **Chapter 15**

We've come to the final steps of designing a Web site with Panels—theming. This development phase puts the last touches on a Web site and brings everything together. Chapter 15 covers styling that can be done from within the Panels UI. You also learn how to apply custom CSS selectors that you can hook into from your own CSS files. Prepare yourself to be amazed at the level of control Panels gives you for theming your site.

## **Chapter 16**

Once a site is built, it must be deployed and made available for use. In Chapter 16, we touch on some of the challenges and changes that come with moving a site from testing to production. Views, Panels, and CCK all have the ability to export their structures, giving you the most leverage over site control; in this chapter, you find out how.

## **Part IV—Appendices**

The appendices cover a range of topics you'll need to truly succeed with this suite of modules. Appendix A covers other, relevant modules you'll want to check out when building a site with CCK, Views, and Panels. Appendix B teaches you the "best practices" for reporting an issue. Appendix C gives you an overview of the plugin classes that are available to programmers in the Views API.

# Creating Views

In this chapter, we talk about the Views user interface (UI) and discover how each function works. We look at how each piece creates part of a query, and how the results of those queries fit into pages and blocks. We also discuss the most important filters you may need and explore how to create relationships between node content that does not otherwise share information. RSS, styles and fields, and the Views Bonus pack are other important topics when determining what you want out of your view.

## Views UI

The Views 2 UI is significantly different from the original Views UI. The rewrite for Drupal 6 allowed for a complete change of the Views control page, bringing the power of AJAX to the Views creation system. The new interface is cleaner, is packed with a large number of features, and gives a distinct clarity to each piece of functionality.

Users of the original versions of Views had to deal with arrows that moved parts of the page around, fields that looked like they might work together but really didn't, and overall an interface that just wasn't very clean. Moreover, when building a query, users really couldn't get a good idea of what the view would ultimately look like unless it was out where other users could see it publicly. This could lead to some ugly or confusing pages temporarily, until the view was completed. For the most part, Views represented a huge step forward over having to custom build your own queries all the time, but there was definitely significant room for improvement.

Several months were spent in the design of the Views 2 UI. The revised UI is compact and clean, allowing users to easily see exactly which part of the view is currently being worked on, which changes have been made, and, most importantly, what the finished view will look like.

When Views is installed, it creates a new menu item under Administer >> Site building >> Views. This menu item is the base Views page. The Views UI consists of four main pages: List, Add, Import, and Tools.

---

### Advanced Help

If you don't have the Advanced Help module installed, now is a really good time to add it to your system. Views takes advantage of Advanced Help to give tips and suggestions throughout the UI. It creates a small pop-up window that contains information about the page you are working on, allowing you to keep the working page available. Look for small circle icons containing a question mark; each will take you to a specific topic in the help system.

Advanced Help is not required to use Views, but Views will provide warning notices if Advanced Help is not installed.

---

There are nearly limitless applications in the real world for Views. Any time you need a list, Views is there. In regard to the examples introduced in this book, you could use Views to create a list of the most recent recipes entered, the most recent batches brewed, shirts available in a particular size, and so on. If you add pictures of a glass of your homebrew, you could use Views to display only the batches for which you have uploaded pictures, and more.

### List

The landing page for Views is the List page. On this page is the list of default views provided with the module install. By default, all of these views are disabled, allowing the administrator to determine which views, if any, might be needed immediately. It also prevents new items from displaying on pages where they may not be expected; the frontpage view, for example, will override your front page. There is also a link to the Getting Started page. It is an example of Advanced Help, and will guide you through the creation of a simple view. Figure 7-1 is an example of what your default Views landing page should look like.

Once you have created views of your own, they will also be listed here in alphabetical order. This list can be modified through the group of drop-down menus found at the top of the list. These drop-downs consist of filters and sorts for the Views themselves.

The first row of list drop-downs are the filters. They filter out any Views that do not match the criteria given:

- Storage: Filters on whether the view is local and in your database (normal), whether default views are stored only in code (default), or whether both code and database are used for storage (overridden).
- Type: Different than the content type. This filter narrows down which sort of content pieces are contained in the view.
- Tag: Lists all tags that are available for currently available views.
- Displays: Lists which type of content sections views can be displayed in.

**Views**
List
Add
Import
Tools

Not sure what to do? Try the "Getting started" page.

**Storage:** <All>
**Type:** <All>
**Tag:** <All>
**Displays:** <All>

**Sort by:** Name
**Order:** Up
Apply

<b>Default Node view: archive</b> (default)	Enable
Path: archive <i>Block, Page</i>	Display a list of months that link to content for that month.
<b>Default Comment view: comments_recent</b> (default)	Enable
Title: Recent comments Path: comments/recent <i>Block, Page</i>	Contains a block and a page to list recent comments; the block will automatically link to the page, which displays the comment body as well as a link to the node.
<b>Default Node view: frontpage</b> (default)	Enable
Path: frontpage <i>Feed, Page</i>	Emulates the default Drupal front page; you may set the default home page path to this view to make it your front page.
<b>Default Node view: glossary</b> (default)	Enable
Path: glossary <i>Page</i>	A list of all content, by letter.
<b>Default Node view: taxonomy_term</b> (default)	Enable
Path: taxonomy/term/% <i>Feed, Page</i>	A view to emulate Drupal core's handling of taxonomy/term; it also emulates Views 1's handling by having two possible feeds.
<b>Default Node view: tracker</b> (default)	Enable
Title: Recent posts Path: tracker <i>Page</i>	Shows all new activity on system.

Figure 7-1 The Views List page

Two types of sorting criteria may be applied in Views. Sort by creates the list by a single particular piece of information about the view, while Order creates an ascending or descending list ordered by the rest of the drop-down choices.

Note that multiple filters as well as both sorting options can be used to change the list of views, narrowing down that list considerably. This makes a case for sites with large numbers of views in use to tag each view with keywords that make it easier to find; this approach allows the Tag filter to be used to minimize the initial returned list. Clicking the Apply button causes your sorts and filters to be executed, returning the list you requested.

The list of default views available is intended to simulate some of the most basic tasks a new Drupal site may need to accomplish. Each view gets a box of its own, displaying the pertinent information about that view. Each view can have many pieces of information, most of which can feed back into the filters.

The next thing to focus on in the Views list page is the title bar for one view. This darker bar contains four pieces of information. In italics is the storage level of the view—normal, default, or overridden. Next is the type of view, which is listed in a normal font and affects the Type drop-down menu. Third is the emphasized text, which is the actual view name. After that, in parentheses, is the tag that is used for that particular view. At the end of the bar is a group of links: Edit, Export, Clone, Delete. For a default view, only the Enable link will be available until the view is enabled. For views that you create, all four links are present at all times. You cannot currently disable a view that you have created.

Inside the box itself are up to four more pertinent Views clues to serve as differentiators for each view. The left column may show the Title, Path, and Display, assuming they have been configured for that view. Title is the human-readable name, much like the Display Name for a field. Path defines the Drupal-specific URL that would take you to a page that displays this information. In italics is the display type, which may contain block, feed, page, or date browser. On the right side of the box is a short description of the purpose of the view or its functionality. This area may also provide prerequisite information on which modules need to be enabled to provide the view with data.

## Add

The Add pages are where all of your view definition takes place. This self-contained, multi-step process keeps the majority of your work on the same page so that you can easily see which parts of the view definition you have changed. The Views UI gives clues as to what has or has not been changed in the display. In the default display for a given view, when a change is made, the changed setting is highlighted and shown in boldface until the changes are saved. Note that if any of the form settings are open for editing in the view, the save button for the overall view will be disabled until the edit itself is saved or cancelled. These pages are also where all of your edits for an existing view will take place.

More clues come at the end of the submenu bar, which contains an italicized note stating whether this is a new or a changed view. This message persists until the changes are saved. The submenu bar is explicit, telling you exactly which view and display type you are editing. At the end of the bar are links to export the view, to clone the view, and, if you are editing, to look at the page in which the view will appear. Exporting a view will give you the code to import the view into another installation. Cloning a view is useful for experimentation. For example, you might want to try some different settings on a view; cloning will let you do that while leaving your original view intact. Cloning also gives you a base to work with if you want to create multiple similar views. For instance, you may have a view of recent comments but also want a view with recent comments for a particular node type. You might clone the recent comments view rather than re-creating it.

## Exercise 7-1

### Enabling and Changing the Default View

In this exercise, we'll enable a default view and make a small change to become familiar with the Edit page.

1. Navigate to Administer >> Site building >> Views.
2. Scroll down to the Glossary view.
3. Click the Enable link on the right side of the title bar.

It will take a moment for the view to become enabled. When it does, the page will refresh and the view will move up in the list and into the enabled views, which are sorted alphabetically by default.

4. Click the Edit link next to the view (Figure 7-2).



Figure 7-2 Edit: one of the available options for an enabled view

5. Under the "View Settings" box, click the word "default" next to "Tag."
6. Scroll down the page to the "View details" box (Figure 7-3) and change the word "default" to anything else. You can change the tag or remove it all together.

 A screenshot of a "View details" form. It has a section for "View description:" with a text area containing "A list of all content, by letter." and a note: "This description will appear on the Views administrative UI to tell you what the view is about." Below that is a "View tag:" section with a text field containing "default 1" and a note: "Enter an optional tag for this view; it is used only to help sort views on the administrative page." At the bottom are "Update" and "Cancel" buttons.

Figure 7-3 Changing the tag for the view



7. Click the update button.

Your change has not yet become permanent; Views allows you to make multiple edits before saving the view. It will let you know that you've made changes.

Figure 7-4 shows you what your submenu may look like after you have updated but not yet saved the changes to the view.

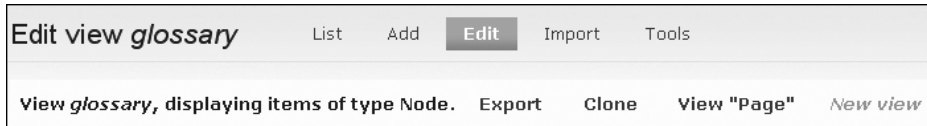


Figure 7-4 Changing the tag for a default view turns it into a new view

You can scroll down to the bottom of the page for a look at the preview; we'll discuss this feature later in the chapter.

8. Click the Cancel button at the bottom of the edit section. We don't actually want to change the tag.

---

When you add a display type, all settings appear in a lighter, italicized font until changes are made. If a change is made to a setting so that it differs from the default, that setting will then appear in a normal-weight, non-italicized font. This system enables you to quickly identify exactly which parts of the view have been modified. The “gear” icons indicate styling options are available for that setting. Some boxes for settings have a plus sign (+) in the box title bar (as seen in Figure 7-5); clicking it automatically opens a corresponding add menu that allows you to select and configure a new option for that set of configuration options. For example, click the plus sign next in the Filters title bar to add a new filter to your view. In the same title box, a small up/down arrow icon will appear when two or more options are available in that section. Click this icon to rearrange the items within that group of settings.



Figure 7-5 Icons used to add, rearrange, and configure parts of a view

## Views Add Creation Page

The Views Add creation page is the first page you encounter when you are creating a new view. It requires two pieces of information, view name and view type, and suggests two others, view description and view tag. The view name can be a combination of alphanumeric characters and underscores; names should use all lowercase letters for consistency. The view description is a text field where you should enter a sentence or two to describe the purpose of the view. A view tag is useful for grouping or separating your views if you are using them for specific topics or purposes. Finally, each view must have a type that determines which content it will look at to create the view. The type determines the primary table that Views will access for data.

Here are the types of views:

- **Node:** Node views are likely to be the most common type of view on a site. The node type creates lists of any node content.
- **Comment:** Comment lists can be retrieved from nodes, but there may be times when you want only comment information. The view may be faster, but there is a tradeoff in security.
- **File:** File limits views to information about uploaded files. It is useful for seeing which files are taking up space in your installation.
- **Node revision:** This limits the view to revision information—useful for determining which users are making changes, and which changes are being made.
- **Term:** Use this view type with taxonomy; it helps to create lists of taxonomy terms.
- **User:** This type breaks down user information and helps with showing which users have accounts, and what they are doing.
- **Access log:** This view can show what your users are accessing, and what they are having trouble accessing. An access log view can help filter down the log so it becomes easier to spot site problems.

Once a view is created, the type of view cannot be changed. If the basic definition is not correct for your needs, the view will have to be re-created. Choosing a particular type means that only certain filters, fields, and other Views functionality will be available once that type is chosen. Figure 7-6 shows the form used to create a new view.

## Left-Side Tabs

Along the left side of the UI is a group of tabs (or drawers). These tabs show which displays the view currently has configured and give you the ability to add new displays. A *display* is a place or way the view may be shown to users of the Web site. All views have a default display. The default display contains the initial and any edited settings for that view. The default display is not actually used anywhere within Drupal itself—something that may confuse new users. It simply is a container, the initial state of the view.

Views
List
Add
Import
Tools

**View name:** \*

This is the unique name of the view. It must contain only alphanumeric characters and underscores; it is used to identify the view internally and to generate unique theming template names for this view. If overriding a module provided view, the name must not be changed or instead a new view will be created.

**View description:**

This description will appear on the Views administrative UI to tell you what the view is about.

**View tag:**

Enter an optional tag for this view; it is used only to help sort views on the administrative page.

**View type:**

☒ Node  
Nodes are a Drupal site's primary content.

☐ Comment  
Comments are responses to node content.

☐ File  
Files maintained by Drupal and various modules.

☐ Node revision  
Node revisions are a history of changes to nodes.

☐ Term  
Taxonomy terms are attached to nodes.

☐ User  
Users who have created accounts on your site.

The view type is the primary table for which information is being retrieved. The view type controls what arguments, fields, sort criteria and filters are available, so once this is set it **cannot be changed**.

Next

Figure 7-6 Creating a new view

Four display types are used for actual visual placement:

- Page: Displays the view as an entire page complete with a menu and URL.
- Feed: Helps set up the format for an RSS feed.
- Block: Creates a view that will be placed within a block.
- Attachment: Helps to add a view to another view.

The first tab will be the default setting for the view; if you are not sure which display you are editing, the main window will provide this information. Figure 7-7 shows an

View archive, displaying items of type Node. Export Clone View "Page"

**Defaults** ▶ **Defaults** *Default settings for this view.*

Page  
Block

Page ▼

Add display

Analyze

**View settings**

Tag: default

**Basic settings**

Name: Defaults  
Title: None  
Style: Unformatted  
Row style: Node  
Use AJAX: No  
Use pager: Yes  
Items per page: 10  
More link: No  
Distinct: No  
Access: Unrestricted  
Exposed form in block: No  
Header: None  
Footer: None  
Empty text: None  
Theme: Information

**Relationships** + ↕

None defined

**Sort criteria** + ↕

Node: Post date desc

**Arguments** + ↕

Node: Created year + month  
Style: List

**Filters** + ↕

Node: Published True

**Fields**

The style selected does not utilize fields.

Figure 7-7 The default display for the archive view

example of one of the views included with the Views module. It has been edited; yours may not look exactly the same.

New displays are created by using the drop-down menu and can be an attachment, block, feed, or page. Adding a display allows you to refine the view for that type of display. For example, you may wish to use a view within a block; you could create a display setting specifically to format for a block. You could then create a page display and use the same view, but format it differently to take advantage of the greater space allotted to a page. Choose a display type you want to add, and click the “Add display” button.

The last drawer is the Analyze button. At this time Analyze has only a minor level of functionality, but like the rest of Drupal, it is very pluggable and can be expanded by those developers who wish to do so. Its purpose is to do a low level of error checking on your view to see if anything is obviously wrong. The Analyze feature was originally added to ensure that a view has content that will display. Some members of the Drupal community suggested that a view should contain a filter that required the content being displayed to be published. There are reasons why this approach is not necessarily a good idea, so the filter is not present. Analyze will notice when you have not set a filter that displays some amount of data and warn you of this potential problem.

## View Settings

View settings is the first box in the first column of the main display. It appears only in the default view settings, as it is pertinent to the entire view for all display types. This box contains the Tag field, which can be edited to add or remove tags even after the

view is being used. To change the tag, click on the tag itself—this will allow you to change both the tag and the description of the view.

### Basic Settings

The basic settings are the first level of definitions. They define the core of what your view will look like, and may occasionally have slightly different effects depending on the display type. Notice the italicized lettering for the display in Figure 7-8. Most settings for a display will be italicized unless you modify this default behavior.

Name is the name for this particular display. Only the administrative interface sees this name.


**Basic settings**  
Name: Page  
*Title: None*  
*Style: HTML List*   
*Row style: Fields*   
*Use AJAX: No*  
Use pager: No  
Items to display: 15  
*Distinct: No*  
*Access: Unrestricted*  
*Caching: None*  
*Exposed form in block:*  
*No*  
*Header: None*  
*Footer: None*  
*Empty text: None*  
Theme: Information

Figure 7-8 Basic settings box for a page display

Title is the displayed title of the view itself and will show wherever the view is. It will be the block title in a block view, or the title of the page if the view is an entire page. You may leave it as “Title” for a default view, but you may want to name it if it is a display, or if you are using a cloned view and the title will be the same wherever it is used on the site.

Style determines how the view itself will actually appear; it does not affect the rest of the page. Your view can be shown in a grid, list, table, or unformatted. Any style other than unformatted will be associated with further settings that customize that style; these options are reached by changing the style or by using the gear icon. Choosing a style determines the look of the view. Each style can be formatted in different ways, from completely unformatted, to a basic list to a neatly stacked grid. Each style type has the ability to group the output by one of the fields that are being displayed; if the post date is in the view, grouping by that date will gather all content added on the same day into one group. Also note that for styles (and for row styles), you can change the look by using template files to specify exactly what you want—more on that topic when we talk about theming in Chapter 9.

The grid style allows you to choose a number of columns that will appear on the view. You may choose to align your view horizontally (top left to bottom right) or vertically (column A from top to bottom, column B from top to bottom).

Lists can be either ordered or unordered. Ordered lists will number the view results. An unordered list provides output similar to the unformatted style. It is significantly cleaner and easier to read than an unformatted view, however, thanks to the bullets that appear next to each entry returned.

Tables are the most complex output format. The table display is clean and easily understandable by most users. By default, each column in a table display contains one field. Columns can be changed to use multiple fields if this formatting is desired. In the style options (which become available when you click the gear icon), choose a field. In the Column drop-down menu next to that field is a list of all other fields available—assuming you have already added fields. Change the drop-down choice to one of the other available fields. This step can be repeated so that all of your fields are in the same column, but only the main column item can be sorted. When multiple values appear in the same column, the Separator field can be used to distinguish between the fields in the column. Separator may use regular characters or HTML. If multiple values are being used, it is highly recommended that you use the `&nbsp;`; (single space) for separation, if nothing else. Bullets and the pipe (`|`) character are also common and valid separators, although it is recommended that `&nbsp;` be placed around these characters for additional readability.

Tables can also be sorted by column, which creates a clickable header. You may choose a column to be the default column on which the table sorts and specify whether the sort is ascending or descending.

The unformatted style provides a simple, basic list of all items that match the query. Each row is a returned field, but lacks any special spacing or styling. This style is potentially useful for lists of names, but may be hard to read for many users if multiple fields are returned.

Row style determines how each row in the view itself will be styled—as a node or as a field. If the Style setting is using Table, Row style will be missing from the Basic

settings box. Row styles may use fields or nodes, with each type displaying one per row. When using fields, you have the option of making each of the fields appear inline rather than stacked, and you can provide a separator just as the full view style does. The node style is exclusive to using selected fields (the fields box), and gives you the option of showing just the teaser instead of the full node. In addition, you can put comment links or comments themselves in the view.

AJAX may be available for use in some cases. The “Use AJAX” option specifies whether you will use AJAX in the view for exposed filters, table sorting, and paging. Be aware that using AJAX will keep the entire page from refreshing, which may cause issues with links.

Pagers make it easy for your users to skip forward and backward in your view, which is a very useful ability within large views. Two types of pagers are available, should they be needed. A mini-pager shows the current page of total pages and uses forward and backward arrows (<< 1 of 6 >>). Mini-pagers are well suited for use within block displays because they fit more cleanly than a full pager. A full pager displays a list of page numbers plus first, previous, next, and last options so that users may easily jump multiple pages. Pager element is a number that can be used to identify a pager within each page. If multiple pagers are used within a page, each one needs to have a different identifier.

Every view needs to have a number of items to return by default. The “Items per page” option sets this number, and can provide a helpful limit. Using a set number can increase the return time of a page—something most Web sites see as a critical function. You can also offset or skip some number of items in your view.

Views can have multiple displays, which may be linked together easily by using the More link. Checking this box will add a More link to the bottom of the view. On a block display, such a link will take the user to the page display version of that view, which may contain significantly more data. This capability could be used to link a teaser to a full article, link the archive of this month’s posts to the full archive, or link a short block of recent comments to a full page of recent comments.

Distinct adds the SQL statement `DISTINCT` to your query, which attempts to remove any duplicated records from the view. Completion of the Distinct operation takes time, which may be considered a performance problem in some cases.

Access places restrictions on what users can see the view, either by user role or by general permissions. You can create roles for users or change user permissions under Administer >> User management >> Roles or Permissions.

Caching is one of the newest features to Views. This pluggable cache is *not* the same as the overall Views caching. Rather, it lets you cache each display separately if you wish, or cache the entire set via the default. The standard options are the ability to cache the query results and/or the entire rendered output. You may cache one and not the other. In addition, you may assign different lengths of time to each cache; the menus provide options ranging from 1 minute to 6 days.

The Header adds a header to the view. A header is simply some text that will appear above the view. It can be set to appear even if the view has no results for display, which helps let the user know that the page did display, even if no actual data was returned. The Footer is essentially the same as the header, except that it appears at the bottom of the

view. Either of these elements can be used to provide an explanation of the view or other pertinent information about the view.

Empty text is text that can be displayed if the view returns no results. It lets the user know that the view has completed its processing. Such text is more useful than just returning an empty result, and is clearer than using just a header or footer. In essence, this text lets the user know that the query mechanism did not crash; there are simply no results.

Finally, there's the theme. Theme is not actually a setting; it is informational in nature. It lists template files that the different parts of the view may be using. This display is similar in nature to what you would see if you looked at a page with the Devel module. The template in use by each piece of the view appears in bold font, so you can see exactly which files are being used. You may have display, overall style, and row style themes within the same view. You can also rescan the template files in case you make changes to the names of `.tpl` files. Clicking the "Rescan template files" button will clear the theme registry.

## Display Specific Settings

Each display type also has a settings box in the first column that appears below the Basic settings box. Figure 7-9 shows one example of display settings.

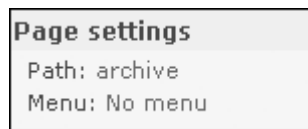


Figure 7-9 Display-specific settings for the archive view

Page has two settings, path and menu. Path is the Drupal-specific path to the page that contains the view. Remember that this path is not a full URL; `http://www.example.com/` is assumed to be the base URL, with additional information being tacked on to the end of this URL. When you click on the path setting, you will see the entire base URL but you must enter the remainder of the path. The path statement can take arguments.

Menu gives you options for adding a menu item as a normal menu entry, a tab, or a default menu tab; you can also not add any menu item at all. When creating a menu item link, you should give it a title and determine whether it should have a weight. The menu also has further options that become available when you click the gear icon, which determine exactly where that menu item should go.

Block has only one setting: admin. This simple description of the block is intended to make it more easily findable in the Administer >> Site building >> Blocks page. The admin setting is a name, and should be relatively short and clear.

If your display type is a feed, two options can be changed. Path is the same as it is for page settings; this time it is the path to the XML feed page, whose filename often ends in `.xml`. The second option is "attach to," which creates the feed as part of one of the other displays. This gives feeds a similarity to attached views.



The settings for attachments also are different from the settings for other displays, as they are not actually changing the display of the attached view. Instead, the attachment settings determine where the attached view will be placed and which information will be inherited from the view it is attached to. Arguments and *exposed filters* may or may not be inherited from the parent view. Exposed filters comprise any of the filters in the view that are made available to the person looking at the view; they can be changed when they are displayed. The attachment can be positioned before the parent view, after this view, or both before and after the view. This group of settings also has an “attach to” option, but it is used to attach the view to one or more displays or to make it part of the default display. Every view built on the default would then have the secondary view attached.

### Relationships

Relationships is the first box in the second column. This and all remaining boxes to set up views have clickable titles. Clicking the title will tell you if that group is using the settings from the default view or if it is using an override. With relationships, you can create links between data that may not be otherwise related. A complex structure can be created by using relationships.

Clicking the plus sign (+) opens the section for adding a new relationship. The different types of content (not content types!) that can be added are grouped together. A drop-down menu allows you to choose which items are available for relationships so that you can more easily find the piece you are looking for. Many groups may be available, and the list changes depending on which content types are available on your site and which other modules you have enabled.

When you add relationships, the number of fields available may increase significantly. For example, you may want users to rate your content. Modules are available to perform this task. However, Views doesn't have a good way to link the various kinds of data together on its own; it needs a relationship. You can create a relationship between voting data and node data, for example, so that they can be displayed together in the list.

### Arguments

The section for adding an argument looks very similar to the section for adding a relationship. The main difference is that many more types or pieces of content can be added as arguments. Arguments act in a fashion similar to filters, but with more exposure to the user. Chapter 8, which is devoted to arguments, discusses this issue in depth.

### Fields

The setup for fields looks significantly like that for arguments and relationships. The fields settings, however, add the actual fields that will be available in your view. This is where you determine which pieces of information will appear in the view and which order they will appear in. Using this feature, you might give a short preview of a group of stories or see a list of recent comments. Fields do not have to be visible to a user to be included in a view.

## Sort Criteria

SQL's sorting mechanisms come into play in the Sort criteria settings. You may sort your view by one or more fields. If you do not specify a sorting criterion, the view will sort itself in ascending alphabetical or numerical order on the first field available to the view. If you wanted to see content posted by users per day, you would add the sorting criteria User: Name, and then Node: Post date, as shown in Figure 7-10. You may also add the criteria in the opposite order, as they can be rearranged using the rearrangement tool.

**Page: Configure sort criterion "Node: Post date"**

? Status: using default values. Override

**Sort order:**

☐ Ascending

☒ Descending

**Granularity:**

☒ Second

☐ Minute

☐ Hour

☐ Day

☐ Month

☐ Year

The granularity is the smallest unit to use when determining whether two dates are the same; for example, if the granularity is "Year" then all dates in 1999, regardless of when they fall in 1999, will be considered the same date.

Update default display Cancel Remove

Figure 7-10 Adding a sort by posting date

## Filters

Filters can be used to limit the fields returned for your view. Note that this limit is based on actual data, unlike the "items to display" link in the Basic settings box, which puts a straight limit on the number of fields. This area is where some especially important criteria for your views are stored. If you wish to publish a view for general use, any information obtained from nodes will usually be taken from published nodes.

**Warning**

By default, the Node: Published filter is not on. In most cases, you need to add this filter. This group of settings is where the Analyze tool is telling you to go when it cannot find data to publish publicly.

It has been suggested that the Node: Published filter be the default. Security concerns dictate that it not be a default filter, however. This is part of the reason why the Analyze tool warns you that no filter is available for published nodes.

**Overrides**

Overrides are another part of a change from the default setting. If a setting is changed, it may need to be overridden. Each of the clickable box titles (relationships, arguments, filters, sort criteria, and fields) allows you to override the default display. When you click on the title, the box at the bottom of the configuration screen tells you whether you are using the default display. From here, when you update the settings, the update will be pushed to the default display. If you want this change to affect only the current display, you must override the defaults.

**Note**

All of these settings can be overridden only as a group, on a per-group basis. An override will take effect for all relationships, for all arguments, for all filters, and so on. Remember that an override is overriding the default!

This behavior may have unintended consequences. Always double-check your results.

**Validation**

Many of the parts that you can use to define a view require a setting, especially for displays other than the default. If your view does not have the correct settings selected, a red warning box, similar to Figure 7-11, will appear. It indicates which settings are missing or set incorrectly. In other cases, such as when no content is available to the general user base being made available for display, a yellow warning box will appear to inform you of that fact. Until the validation errors are corrected, the live preview function will be unable to display a view.

Display Defaults uses fields but there are none defined for it or all are excluded.  
Display Block uses fields but there are none defined for it or all are excluded.  
Display Page uses fields but there are none defined for it or all are excluded.

Figure 7-11 Validation errors

## Live Preview

The live preview feature represents one of the biggest steps forward in usability for those individuals who are new to using Views. Live preview shows you exactly what your view will look like to your Web site visitors, allowing for changes to be made quickly and easily without needing to click back and forth and constantly reload pages (see Figure 7-12).

**Live preview**

**Display:** Defaults ▾ **Arguments:**

Separate arguments with a / as though they were a URL path.

- ☐ June, 2008 (2)
- ☐ July, 2008 (6)
- ☐ August, 2008 (16)
- ☐ September, 2008 (3)

Figure 7-12 A live preview of the archive view

Other basic information about the view is also presented as part of the preview. This information includes the query that is used to create the view, the title and path, and statistics on how long it takes the query to actually build and execute as well as how long it takes the view to fully render.

## Import

The Import page has a simple purpose: to import a view from another site or installation of Drupal. If you have exported a view, you may enter a new name if you wish, or leave the view Name field blank. Remember that the view name must consist of alphanumeric characters, but may use underscores. Paste the exported view into the text box and click the Import button. Views will then attempt to import the view. If this operation is successful, the Edit page will appear, allowing you to customize the view.

## Tools

The Tools pages provide help with troubleshooting and updating your views to the new version. There are two subtabs within the Tools section: Basic and Convert.

## Basic

The Basic tab comprises a list of check boxes that can be selected to enable features that are helpful for performance optimization, troubleshooting, or general placement of some of the query information within the user interface. This page also contains the highly important Clear Views' Cache button. If you are changing your views, and they are not appearing correctly on your actual displayed pages even though they worked perfectly in the preview, you may need to clear the cache. At the bottom of the page is a drop-down menu that allows you to select where the performance statistics should be placed; this location is entirely up to you.

The check boxes on the Basic tab are documented within the page display itself, and should be easily understood in general. Many of these features are designed to work with the Devel module or to turn off functionality that may be causing problems either for the view or for the browser itself.

“Add Views signature to all SQL queries” tacks an additional field that does not need processing onto the SQL query when it is being built. When looking at an SQL WHERE clause, you will see a 'VIEWS' = 'VIEWS' string that indicates Views was used to build the query. When you are searching through an extensive log file to determine which queries may be causing problems, this string makes a Views query easily identifiable. It is recommended that this flag be used only during troubleshooting, even though it does not appreciably change the query. Why process more than you have to?

Views caching can be disabled across your site with the “Disable Views data caching” check box. This setting may be useful during the creation of new views and especially during the retooling of an existing view. Views may cache a tremendous amount of data in an effort to quickly display a view; this data can come from multiple tables, other modules, or other existing views. Trying to maintain this cache can place a significant burden on performance. If the cache is not used, Views is forced to rebuild each view every time it will be displayed on a page. If that view has to call other views and other modules to return information, things can get very ugly, very quickly.

### Note

Selecting the “Disable Views data caching” check box does not clear pluggable caches (found under the Basic settings); Views does not know where those caches store their data and cannot effectively clear them.

If you need to turn off the pluggable cache, it is best to do so from the view itself while you are developing the Web site. Turn it on when you're ready to deploy a Web page, and always retest the page to make sure it works correctly.

The next two boxes deal with queries and live preview. We've already seen that the live preview shows you what your view is supposed to look like. Normally, your preview will be displayed and the query used to create that view and its corresponding details will appear below it. These settings boxes allow you to put the query above the preview (“Show query above live preview”) and to show all of the queries that are run to create

the view (“Show other queries run during render during live preview”). When troubleshooting, these settings can help you determine if a query is being called multiple times when it should not be.

If you have created a view and have the ability to edit it, you may see [Edit], [Export], and [Clone] links over the top of a view that appears in the public section of your site. These are called hover links. You can turn them off by selecting the “Do not show hover links over views” check box. Hover links are useful for quickly reaching the Edit page for the view directly from the view itself. This is much quicker than digging into `admin/build/views/`, finding the correct view, and hoping you’ve got the right one.

Web site developers should be concerned with the performance of their sites, especially if the site is professional in nature. Views can interact with the Devel module and provide performance and query statistics. Such data helps you determine how long your view and its container are taking to render—again helping you find bottlenecks in system response.

The last check box continues a core tenet within Drupal development—that the entire user interface should work without JavaScript. If you are having problems with using the Views UI because of your browser, or you’re just that concerned about JavaScript security, you can turn off JavaScript for views. The interface should degrade and be completely usable without it; it just won’t be as pretty or as easy to use.

## Bulk Export

The Bulk export page provides a method for exporting all of the code that creates a particular view. Figure 7-13 is part of this page. Quite simply, you can choose one or more views to export, and have the code available and easy to store in the source control repository.

At the bottom of this page is a box labeled “Module name.” This box’s purpose is to help you export a view that you can then easily import directly into a module. If you enter a module name, parts of the code will contain pertinent hooks into the module.

## Convert

The Convert tab is likely to greatly interest users of Views 1. Views 1 is not being converted to Drupal 6, and Views 2 is not being backported to Drupal 5. This leaves users in the unenviable position of having to upgrade their Drupal version as well as Views at the same time to maintain a functioning Web site.

The Convert tool checks the database to see if it contains any Views 1 views and then gives you the opportunity to convert each such view to the Views 2 format. It is highly recommended that you use a test site to do so. Using a test bed to convert the view means that you can then export it and later import to the live site once it is upgraded. You have to do the conversion only once, and it can be perhaps ahead of time; this eases the pain of upgrading.

Basic **Bulk export** Convert

Show only these tags:

Date  
base  
default  
flag  
nodequeue  
recipes  
reviews  
shirts  
testing  
users  
votingapi

Apply

<input type="checkbox"/> Views	Tag	Description
<input type="checkbox"/> Recent_recipes	recipes	Most recent recipes added
<input type="checkbox"/> archive	default	Display a list of months that link to content for that month.
<input type="checkbox"/> backlinks	default	Displays a list of nodes that link to the node, using the search backlinks table.
<input type="checkbox"/> blog_rss	users	RSS feeds for user blogs
<input type="checkbox"/> comments_recent	default	Contains a block and a page to list recent comments; the block will automatically link to the page, which displays the comment body as well as a link to the node.

Figure 7-13 Views Bulk export page

Be aware that due to the extensive changes made between Views 1 and Views 2, the actual views are likely to be altered somewhat by the conversion process. You will need to spend some time examining each converted view to ensure that it still shows (or doesn't show) what is expected. If you have not yet upgraded to Views 2, consider exporting your views from the previous version. Once the upgrade is complete, you can import these views via the Import tab. It will take you directly to the editor and give you a clear idea immediately if the view needs work before being made public once more.

With all of these boxes, check boxes, and settings, it's easy to lose track of where you're going. The next exercise takes you through the creation of a new view.

## Exercise 7-2

### Creating a View: Recent Content

In this exercise, you will create a simple view. Before proceeding with this exercise, make sure you have several nodes created for one of your content types. This example will create a default view for use in displaying the most recent content of all types.

1. Navigate to Home >> Administer >> Site building >> Views and click the Add tab.
2. Use the following values to populate the initial view creation page:

Field	Value
View name	recent_items
View description	Most recent items posted
View tag	base
View type	Node

Notice that this data doesn't actually say "shirts" or identify any other content type specifically. We are creating a more generic view that can apply to all content types—giving us something we can clone later to create a view for a specific content type. For now, we'll use the "base" tag to indicate it's a view we intend to start from to create others.

3. Click the Next button to proceed.
4. In the View settings box, notice that our tag "base" is now shown. To change the tag later, come back to this page and click the word "base."
5. Under the Basic settings, do not change Name or Title. When we are doing a display, we might want to change these settings, but not yet.
6. Change the Style to an HTML list and update the settings. This action will bring up the style settings. The default is an unordered list; leave it alone, and update the settings. Notice one of the red warning boxes at the bottom of the page: Display Defaults uses fields but there are none defined for it or all are excluded.
7. Row styles are set to fields by default. For now, leave this setting as is. Attempting to update this choice will also generate an error message.
8. Skip to the Fields box. Click the + sign to add two fields: Node: Post date and Node: Title (you can access the Groups drop-down menu and choose Node to find these fields more quickly).
9. You will be presented with the configuration screen for Node: Post date. Use these values to configure the field, and then update the settings:

Field	Value
Label	Clear the label—by default, this says "Post date"
Exclude from display	Unchecked (this is the default)
Date format	Custom
Custom date format	F j, Y



10. The UI will immediately take you to the configure screen for Node:Title. Use these values for the fields, and then update the settings:

Field	Value
Label	Clear the label—by default, this says “Title”
Exclude from display	Unchecked
Link this field to its node	Checked

11. Live preview should now be displaying data. Use the up/down arrow on the Fields box, and switch the order of the field display so that Title is first, followed by Post date. Update the settings, and review how the live preview has changed.
12. Click the + next to Sort criteria and choose Node: Post date. Change the sort order to descending and day. These criteria force the most recent posts to be displayed first.
13. Click the + next to Filters. Choose Node: Published. Checking the box on the configure page means that only published nodes will be displayed in the filter for any user.

Your default display is now complete. Your live preview should now display a list of all nodes and dates on which those nodes were published. Each node title should also link directly to that node.



Figure 7-14 An example of the `recent_items` view

Now that you have completed development of a base view, you can clone the view and set filters so that only one type of node is displayed in the view. At this point, we also need to create displays for the view so that it can be placed into Web pages.

## Showing Your Views to the World: Creating Displays

Now that you have a view, you will want to turn it into something usable that you can place on your Web site somewhere for visitors to see. This requires creating a display, placing it into some type of Drupal container, and positioning that container in a particular place. This operation may sound complex, but it does not have to be. We've already talked about the various display-specific settings that you can set during creation or editing of a display. It's now time to look at those displays and put them to use.

Views provides four types of displays: block, page, attachment, and feed. We've talked about the specific settings for those types already, but we haven't delved into the process of using them to create Web pages or parts of Web pages. In the following exercise, we'll create a display using a block—Drupal provides blocks as part of the core functionality.

---

### Exercise 7-3

#### Creating a Block Display

Every view needs a display before it can be inserted on an actual page for use. A block is the easiest type of display to create and understand.

1. Open the view created in Exercise 7-2 (`recent_items`) by going to its Edit page.
2. In the left-hand drawers, change the drop-down choice to Block, and click the “Add display” button. You should now see a new drawer under Default that says Block. Note that if you do not save the view before exiting this page, your new display will be lost.
3. In the Basic settings box, notice the Name field. It says Block, which matches the display type. For the base view that you'll be cloning, you can leave this name alone. For a cloned view (e.g., for Joe's Shirts), you might want to name it something more specific, such as `Block_shirts`.
4. Change the display-specific setting if desired. You can give the block an administrative title, making it easier to find in the blocks administration page. You can also set up caching here for the block. Note that this is Drupal core's block caching, not Views based.
5. Save your changes.

Now you have a display that can be shown as a block. It can be administered from the Blocks page.

---

With the creation of a base view and block display, you are ready to clone that view for use and make it exactly what you want with only a few modifications, rather than having to re-create the entire view every time you want to customize it.

## Exercise 7-4

### Cloning a View to Create More Specific Content

Cloning a view creates an exact duplicate of an existing view. You are required to give a new name to the cloned view to ensure its uniqueness, and you are allowed to change the description and tag. If you haven't created any entries for Joe's Shirts, please do so before continuing with this exercise.

1. Open the `recent_items` view.
2. Choose the Clone item in the menu bar.
3. You will be taken to the Add screen. Change the view name to `recent_items_shirts`.
4. Change the view description to "Most recent shirts posted."
5. Click the Next button.
6. In the Filters box, click the + sign to add a new filter.
7. Choose the filter Node: Type.
8. Choose the Operator "Is one of" and the Node type "Joe's Shirts."
9. Update the settings.
10. Save your changes.

This set of steps creates a view that displays only Joe's T-shirts. The displays from the base view are copied to the new view as well, and are ready to be put onto pages.

---

Now you have a base view and a specific view. Both of these views use the very basics of Views functionality. Of course, having a view defined doesn't do your users any good if it's not shown to them. At this point, we need to create a place for the view to be contained. One of the easiest ways to do so is to create a block.

## Blocks

Blocks are a core Drupal containment system. It's necessary to know about them to use Views effectively, although we cannot hope to cover every use of blocks within this book. Blocks are one of the most common containers for content in Drupal. They can be placed on your page and hold many types of content. For example, blocks can be easily placed within the left or right sidebars, the header or footer, or the main page content. Even more helpfully, the Blocks page under Administer >> Site building >> Blocks has small boxes on the page to show you exactly which region is which.

The process of creating a block display with Views does not just create a display; it also creates the block for you. Blocks containing views are administered just like any other block in Drupal. In some respects, a block looks very much like a node. Blocks were originally used for customization purposes—using them was an easier way to

embed custom PHP code into a Web site. As time has passed, this characteristic has become less useful; PHP embedded in blocks is not easily upgraded, nor is it stored in source control. Source control is critical for many organizations, especially those that are doing serious code development work.

To embed a view into a block using the UI, you must create a block display within the Views interface. The blocks UI does not allow you to import a view into a block.

---

## Exercise 7-5

### Working with Blocks

In this exercise, we will place a block that contains a view in the right sidebar of the page.

1. Navigate to Administer >> Site building >> Blocks.
2. Review the list of available blocks. In the lower section of the page is the Disabled blocks list. Locate the `recent_items_shirts` block.
3. Use the drop-down menu options to choose a placement *or* use the grab handle to drag the block into the region where you want the block to appear. For this exercise, choose the right sidebar. If you use the drop-down menu, the block will immediately jump to the selected region.
4. Save the block. If you forget to save the block and continue working, the block will remain disabled.
5. Choose the configure link to the right of the block name.
6. Change the block title to “Newest shirts!”
7. Save the block.

Observe your page. You should see a section in the right sidebar that contains your new block, and a list of the most recent shirts added.

---

Using this type of block view can be very useful. Blocks can be configured so that they are shown only to certain users, only to users who are authenticated, only on certain pages, and so forth. With just a few small changes, you could create two similar views and place them in the same location, where site visitors would see a view with an article name and a teaser, but authenticated users or subscribers could have that same view link to a full article.

## Pages

A view can be created as an entire page. We’ve already created a block view for Joe’s most recent items, and one for just his shirts. Users may want to see a full list of shirts, however, and the block view can link to the full page view. The page display automatically creates an alias that can be used in the URL to take you directly to the page containing that view.

Pages are a place where overrides really start coming into play with Views, and it becomes very important to understand exactly how they work. In a block display, a limited amount of information should be presented; presenting too much information crowds the block and overwhelms the user. However, a full page view is where users expect to get a majority, if not all, of the information they are seeking.

## Exercise 7-6

### Creating and Using a Page Display

In this exercise, we will create a page display to go along with our recent shirt block. We want to allow users to see all of Joe's shirts, so we'll need to let them page through the list. We might also want to take a look at the formatting to see if another option would be cleaner for displaying this information on a full page than a basic list.

1. Navigate to Administer >> Site building >> Views and choose Edit next to `recent_items_shirts`.
2. In the default display, under Basic settings, change the More link to "yes," if it is not already set.
3. Change the Items to display to 5; now only five items will be shown on the page.
4. Save the view.
5. In the "Add display" drop-down menu, select Page. Click the "Add display" button.
6. Now you will have a page display available. Highlight the page display so that you can make changes to it.
7. Under the Basic settings, Items to display is set to 5. Click the 5, and look down to the box. Change the setting to 15, and click first Override and then Update. Now, for this display only, 15 items will be displayed. Notice that the status changes to "using overridden values" and the button changes to "Use default," as seen in Figure 7-15.

**Page: Items to display**

❓ Status: using overridden values. Use default

15

The number of items to display per page. Enter 0 for no limit.

**Offset:**

0

The number of items to skip. For example, if this field is 3, the first 3 items will be skipped and not displayed. Offset can not be used if items to display is 0; instead use a very large number there.

Update Cancel

Figure 7-15 Configuration box changes after overriding the default

8. Now that there are more items on the page, it might be useful to add a pager. Change the Use pager setting to “Full pager.”
9. Under Page settings, change the Path variable to “shirts” and update the settings.
10. In the Fields box, you may choose to add extra fields—you may have a purchase price, an image of the shirt, comments about it, and so on. If you add any fields, click Override.
11. Save the view.

If you navigate to <http://www.example.com/?q=shirts> now, you will see a full page of shirt content. If more than 15 items are available, a pager will appear that helps you to navigate through all of the pages. Thus your users can scroll through all the available shirts easily. If you have enough shirts to pass the block’s display limit, your block will display a “More” link that takes you directly to the shirts page.

---

## Attachments

Attachments, put simply, embed a view within another view. The glossary view is the most obvious example of an attachment. Within the glossary view, the page view is the list of nodes; along the top is the attached view, which summarizes how many nodes begin with a particular letter and number.

Another use for attached views is to embed an archive of recent activity within the content page. Most archive listings appear in sidebars or on their own pages. This format represents a change from the more typical river of news style employed by many sites.

Attachments can inherit arguments from the view they are attached to. This capability gives you the ability to filter the entries displayed.

## Feed

RSS feeds are the most common way to notify people of new content on a site. Most blogs use a feed, which lets their readers follow the blog in their choice of blog reader, lets readers easily keep track of which posts have been read, and, most importantly, lets readers easily get information pushed to them rather than forcing readers to seek it out.

Creating a feed view is almost as straightforward as it gets. Feeds do not allow for field selection, eliminating that list of choices. The feed style gives you two options: use the site’s mission statement for the feed description or enter your own. Only the Row style: Node can be used with feeds, and the style options are to use the default RSS settings, send the node title, send the title and teaser, or send the full node. The feed display does not override any of the default options. Feeds are not generally user visible and are certainly not easily readable by the user owing to their XML-based output.

## Summary

This chapter covered the basics of creating a view and developing a display. Views has a large number of options and capabilities packed into a small user interface. By employing the power of Views, you can create lists of almost anything you can imagine, in many different fashions.

# Index

## A

---

**abstraction, in object-oriented programming, 186**

**ABV (alcohol by volume), 53–54**

**ABW (alcohol by weight), 53–54**

**access, Views UI**

access log view in Views Add Creation page, 115

plugins, 193

restrictions, 120

**access rules, Panels UI**

adding access controls, 245–246

overriding core display pages, 247

overview of, 244–245

selection rules as form of, 230

**Acquia Drupal stack, 11**

**Add content**

access controls, 245

arguments in panes, 265

arguments in panes with views, 269

content panes, 238–242

contexts, 259, 261

in Flexible panel layouts, 278

node in Node Reference field as context, 272

node override page, 249–250

node views, 248

relationships, 272

styles to panel regions and panes, 283

taxonomy overrides, 252



**Add content** (*continued*)

taxonomy terms, 263–264

views in panes, 266–268

**Add Content Type page**

adding new types with, 17–18

creating field with complex settings, 30

creating new content type, 13

**Add display button**

adding page display to arguments, 143–144

creating block display, 131

creating page display, 134

using view pane display with argument, 270

Views UI, 117

**Add pages, Views, 112****Add Variant option, Panels UI**

defined, 230

node override, 249

taxonomy override, 252

`add_field()`, query object, 198

`add_groupby()`, query object, 199

`add_having()`, query object, 198

`add_orderby()`, query object, 199

`add_relationship()`, query object, 197

`add_table()`, query object, 197–198

`add_where()`, query object, 198

**Additional Display Settings,**

Date module, 49–50

**add-on field modules, 14****add-on helper modules, 14**

`adjust_join()`, query object, 197

**Admin title, pane settings in Views, 268****administrators**

dashboard, 308

menu links for Panels, 225

**Advanced Help**

Calendar, 308

overview of, 110

reading as documentation for module you are reporting, 310

Views, 208

**Advanced Profile Kit (APK), 308****AJAX, Basic settings in Views UI, 120****alcohol by volume (ABV), 53–54****alcohol by weight (ABW), 53–54****aliases**

execution-related hooks and, 205

for query object, 198–199

referring to given nodes, 5

using in URL to go to page containing view, 133

**Allow settings, Pane settings, 268, 271****Allowed Block content, Panel settings, 227****Allowed values setting, fields, 29, 31, 33****Analyze button, View left-side tabs, 117****anonymous page caching, 182****ApacheSolr module, 306****APIs (application programming interfaces)**

CCK. *See* CCK API

developing CCK with consistent, 8

flexibility of node, 4

Views. *See* Views API

**APK (Advanced Profile Kit), 308****architecture**

data, 185

Drupal core procedural, 186–188

Panels pluggable, 221

Views using object-oriented, 185

`arg()` function, 144–145

argument default plugins, Views, 193, 212

Argument input, Pane settings, 268

**argument validator plugins, Views, 193, 212**

#### **arguments**

- adding, 141–142
- configuring, 139–141
- expanding and stacking, 144–145
- as filters, 138–139
- handlers for, 316
- introducing with Views, 264–265
- not passing to blocks, 144–145
- overview of, 137
- as part of view, 142–143
- using feeds with, 143–144
- in view execution cycle, 201–203
- View UI settings for, 122

#### **arguments in panes**

- Add content modal for views, 266–267
- adding view, 265–266
- overview of, 264–265
- using view pane display with, 270–271
- using with view, 269–270
- view pane displays, 268–269

**Arguments text box, Pane settings, 269**

**ascending sort order, 106**

**attached views, 306**

#### **attachment display**

- defined, 116
- display-specific settings for, 122
- overview of, 135
- user feeds with arguments, 144
- workflow settings, 21

**.attachment-after class, 155**

**.attachment-before class, 155**

**attributes, node, 6–7**

#### **Authoring information**

- adding new nodes, 235–236

changing using Views Bulk Information, 307

as node content, 6

overriding core node editing pages, 253

query displaying, 107

using relationship for node, 271–272

**autocomplete widget, 45–46**

---

## B

---

**backup, site, 292**

#### **base tables**

- concept of, 183
- declaring tables in `hook_views_data()` as, 208–209
- overview of, 191
- relating tables to each other, 206–207
- and relationships, 191–192

**base theme, preserving, 60–61**

**Basic page, Display fields, 34–35**

**Basic settings, Panels UI, 230**

#### **Basic settings, Views UI**

- creating page display, 134
- determining themes views are using, 156–157
- overview of, 118–121
- user feeds with arguments, 144

**Basic tab, Views UI, 126–127**

#### **block displays**

- creating, 131
- creating blocks with, 132–133
- defined, 116
- display-specific settings for, 121
- mini-pagers suited for, 120

#### **blocks**

- not passing arguments to, 144–145
- overview of, 132–133

**blocks** (*continued*)

pushing into page template, 218  
working with, 133

**blogs**

Blog entry content type, 5  
`blog_rss` view, 309  
taxonomy used in, 262

**body field, of nodes, 7, 20****Book outline settings, panel nodes, 235****Book page content type, 5, 36****book parents, 272****breadcrumb trail**

arguments modifying, 202  
changing in Panels dashboard, 226  
editing mini-panel using, 237  
executing views display, 203–204  
how it works, 30  
Node Reference tool and, 42  
wildcards and, 140

**bugs, reporting issues, 312****`build()` method, view execution cycle, 202–203****Bulk Export module, CTSuite, 222****Bulk export page, Views UI, 127**


---

## C

**caching**

Clear Views' Cache button, 126  
clearing if field settings change, 98  
creating data for fields, 95  
CSS, 285  
disabling pluggable, 126  
disabling Views, 126  
for each view display, 120  
hooks to improve performance, 201  
panels, 221, 242  
query optimization with, 181–182  
saving views, 142

**Calendar module, 307–308****Canvas menu, Flexible panel layout, 277****Cascading Style Sheets. See CSS (Cascading Style Sheets)****Category, Pane settings in Views, 268****CCK (Content Construction Kit) module**

adding fields with, 7  
Calendar module, 307–308  
creating new content type, 11–15  
defined, 1  
development of, 8–10  
Drupal version numbers and, 10  
exporting, 295–296  
forum on status of, 9  
getting started with, 10–11

**CCK API**

creating data for fields, 95–97  
creating field instances using Content Copy, 91  
creating field instances using CRUD API, 91–95  
field model. *See* field model  
formatter modules, 87–91  
helper functions, 97–98  
using, 75–76  
widget type modules, 85–87

**`/cck/theme` directory, 59–60****Center region, Flexible panel layout, 277****Chaffer, Jonathan (JonBob), 9****Chaos Tool Suite. See CTools (Chaos Tool Suite) module****check boxes/radio button widget, 45–46****`check_markup()` function, Contemplate, 72–73****`check_plain()`, Contemplate module, 72–73****`check_plain()`, `dsm()` function, 167**

**Choose Layout option**

- node override, 249
- taxonomy override, 252

**classes**

- in Panels-based CSS, 286, 289
- in Views, 153–156, 163–164, 318–326

**Clear Views' Cache button, 126****Clone option, Panels UI, 229, 260****cloning views, 112, 132****code flows, handlers, 210****Column menu, Flexible panel layout, 277****columns, table style of view, 114****comments**

- for content types page, 22–23
- for node contexts, 261
- for node overrides, 250
- for node view page override, 248–249
- for panel nodes, 235
- template for, 164
- view, 115

**Computed Code field, 51****Computed Field module, 51–54****construct( ) method, 195, 201****constructors, object-oriented programming, 190****Contemplate (Content Templates) module, themes, 71–73****content**

- adding to panels, 237–242
- changing layouts and, 279–280
- configuring development environment, 291
- node override, 249–250
- tagging to build taxonomy, 262
- taxonomy override, 252

**Content Construction Kit. See CCK (Content Construction Kit) module****Content Copy module, 14, 91****Content module, 14****content panes**

- adding, 238–242
- configuring existing, 243–244
- enabling views on Panel dashboard as, 226–227
- introducing arguments with, 265

**Content Permissions module, 14****Content Profile module, 272****Content setting, Variants tab of Panel UI, 231****Content Templates (Contemplate) module, themes, 71–73****content types**

- adding fields to, 24–29
- advantages of CCK, 9–10
- creating fields for existing, 27–29
- creating new, 11–15
- creating new, with Flexinode, 8–9
- default, 4–5
- Drupal basic, 4
- limiting links to specific, 46
- sharing fields between, 29
- what to know before creating, 17

**content types page**

- Comment settings, 22–23
- creating new content type, 11–12
- Export tab, 22
- Fields tab, 22
- Identification settings, 18–19
- Import tab, 22
- overview of, 17–18
- Submission form settings, 19–21
- Workflow settings, 21–22

**\$content variable, 68****content\_clear\_type\_cache( ), 98****content\_database\_info( ), 97**

content\_field\_instance\_create( ), **91–92**  
 content\_field\_instance\_read( ), **94**  
 content\_field\_instance\_update( ), **94**  
 \_content\_field\_invoke( ), **77**  
 content\_fields( ), **97**  
 content\_format( ), **97**  
 content\_write\_record( ), **98**  
 content.crud.inc, **92**

**contexts**

adding, 260–262  
 creating in panels, 221  
 designing page content/layouts with, 280  
 overview of, 257–258  
 in Panel pages, 258–259  
 taxonomy in, 262–264  
 taxonomy override, 251–252  
 User Reference and Node Reference, 272–273

**Contexts setting, Variants tab of Panel UI, 230**

**Convert tab, Views UI, 127–128**

**copying template files, 156**

**COUNT function, query object, 198–199**

**CPU time, development time vs., 177**

**Create new revision option, Workflow, 21–22**

**Create Panel screen, panel nodes, 234–235**

**Create Variant**

node override, 249  
 taxonomy override, 252

**CRUD (Create, Read, Update, Delete) API, for field instances, 91–95**

**CSS (Cascading Style Sheets)**

caching, 285  
 creating new panel style, 282–283

grouping fields in template, 172–173  
 identifying particular pane for styling, 286–289  
 in Panels UI, 284–285  
 setting properties within panels using Stylizer, 280  
 in source code, 285–289  
 theming a fieldgroup, 66–68

**CTools (Chaos Tool Suite) module**

contexts. *See* contexts  
 downloading, 222  
 history of, 218  
 managing access rules with Page Manager, 244–246  
 overview of, 223

**Custom content panes, CTSuite, 222**

**custom pages, 228**

**Custom rulesets, CTSuite, 222**

**Customize date parts, Date module, 49**

**Customize Default Value, relative dates, 48–49**

---

## D

**data**

creating for CCK fields, 95–97  
 Views API architecture, 185

**data types**

CCK, 26–27  
 choosing field for, 39–43  
 converting in development site, 294  
 for field in Type fields, 25  
 relational database fields, 103  
 widgets for, 44–47

database columns **operation**,  
 hook\_field\_settings( ), **80–81**

**database schema**

declaring fields on tables, 209–210

- declaring tables in
    - `hook_views_data()`, 207–209
  - overview of, 206
  - relating tables to each other, 206–207
- database storage, 8–9**
- Date module**
  - Date field, 50–51
  - defined, 14
  - overview of, 48–49
  - required by Calendar, 308
- Datstamp field type, 48**
- Datetime field type, 48**
- `db_query()`, **query object, 198–199**
- `db_rewrite_sql()`, **view execution cycle, 203**
- Decimal field type**
  - defined, 26, 46
  - working with, 40
- Decimal marker setting, widget types, 46**
- default content types, 4–5**
- default display**
  - creating view, 128–130
  - Date setting, 49–50
  - default template vs., 158
  - in Views UI, 115–117
- default options, Workflow, 21**
- default templates, 156–159**
- default views**
  - enabling and changing, 113–114
  - view execution cycle checking for, 201
  - View settings, 117–118
- `$definition` **array, views\_object, 194**
- Delete operations**
  - content type List page, 18
  - `hook_field()`, 82
  - Panels UI, 229
- Deploy module, for site migration, 299**
- deployment. See site deployment**
- descending sort order, 106**
- DESCRIBE statement, vs. EXPLAIN in MySQL, 180–182**
- Description field, Identification setting, 18–19**
- design, Web site, 14–15**
- `destroy()` **method**
  - view execution cycle, 203
  - `views_object`, 195
- destructors, object-oriented programming, 190**
- Devel module, installing Theme Developer with, 70–71**
- development environment, configuring, 291–293**
- development time, CPU time vs., 177**
- directories**
  - installing modules into, 11
  - themes, 59–60
  - views, 156
- Disable option, Panels UI, 230**
- display**
  - modules in Panels, 222
  - overriding. *See* overriding core display pages
  - Panel content screen settings, 238
  - point-and-click layout in Panels, 219
  - templates, 159–160
- Display Fields tab**
  - adding fields, 24–25
  - adjusting display, 34–35
  - excluding fields, 68
- display plugins, Views API**
  - defined, 192, 212
  - instantiating in initialization, 201
  - list of, 317
- `$display` **variable, PHP, Computed Code field, 51**

**displays, Views UI**

- adding views to panel pane with, 268–269
- Basic settings, 118–121
- block displays, 131–133
- defined, 115
- executing, 203–204
- page displays, 133–135
- of specific settings, 121–122
- types of, 116–117
- using attachments, 135
- using feeds, 135
- using left-side tabs to add, 115–116
- View settings, 117–118
- viewing on Views list page, 111

**DISTINCT statement**

- query object, 197–199
- removing duplicated records from view, 120

**documentation**

- from development to production, 294
- reading for module you are reporting, 310–311

**drag and drop, of existing content panes, 244****Drupal**

- development of CCK, 8–9
- installing for CCK, 10–11
- layouts, 218
- “learning cliff” of, 102
- version numbers, 10

**drupal\_execute( ), 95–96****drupal\_render( ), 88****drupal\_set\_message( ), 167****drupal\_write\_record( ), 98****Drush module, for site migration, 299****dsm( ) function, 165, 167**

---

**E**

---

**Edit operation, content type List page, 18****editing**

- enabling and changing default view, 113–114
- of existing view in Add page, 112
- mini-panels, 237
- overriding core node editing pages, 253–255
- panel content, 238
- panel nodes, 236

**Email module, 55****embedding queries, 179–180****empty text, 121****encapsulation, 186, 189****ensure\_table( ), query object, 197–198****Exclude option, Display Fields tab, 68****execute( ), view execution cycle, 203****execute\_display( ), Views, 203****Existing field, field display, 34****expanding arguments, 144–145****EXPLAIN statement, query optimization, 180–182****explicit relationships**

- overview of, 192
- relating tables to each other as, 206–207

**Export option, Panels UI, 229****Export tab, content types page, 22****exporting structures**

- CCK, 295–296
- Panels, 298–299
- site deployment and, 294–295
- Views, 296–297

**exporting views, 112, 305****exposed filters, 145–147, 163****Extra column, EXPLAIN statement, 181**

---

## F

**feature requests, 312**

**Features module, site migration with, 299**

**feed display**

- adding to arguments, 144
- defined, 116
- display-specific settings for, 121
- overview of, 135

**.feed-icon class, 155**

**feeds, using with arguments, 143–144**

**field handlers, Views API, 315–316**

**field IDs, field names vs., 158**

**field instances**

- creating using Content Copy module, 91
- creating using CRUD API, 91–95
- defined, 17, 37
- overview of, 38

**field model**

- for custom field types, widget types or formatters, 77–79
- field type modules, 79
- hook\_field functions, 80–84
- for new field types, widget types and formatters, 76–77
- overview of, 76

**field modules, 13–14**

**field names, vs. field IDs, 158**

**field templates**

- changing parts of themes, 64–65
- creating themes, 61
- files and filenames for, 63
- variables available in, 63–64

**field type modules**

- hook\_content\_generate(), 84
- hook\_content\_is\_empty(), 84
- hook\_field(), 82–83

hook\_field\_info(), 79–80

hook\_field\_settings(), 80–82

**field types**

- creating custom, 77–79
- creating field instances using, 91
- creating new, 76–77
- defined, 17
- modules defining, 79

**field\_name, creating field instances, 91**

**\$FIELD\_NAME, node templates, 65–66**

**Fieldgroup module, 14**

**fieldgroups, themes for, 66–68**

**fields**

- adding complex settings, 30–33
- adding to content types, 24–26
- adding to content types with CCK, 13–15
- adding to content types with Flexinode, 8–9
- adding to existing content types, 27–29
- adding to nodes, 7
- building SQL queries in Views using, 107–108
- CCK fields composed of, 76
- changing display of, 33–36
- choosing types, 39–43
- Computed Field module, 51–54
- concepts, 17
- for content type pages. *See* content types page
- creating data for CCK, 95–97
- data types, 26–27
- Date module, 48–51
- declaring table, 209–210
- defined, 17
- exporting CCK, 296



**fields** (*continued*)

- filtering and sorting, 105–106
  - identifying insecure, 72
  - Link and Email, 54–55
  - migration of existing settings not supported for, 294
  - overriding core node editing pages, 253
  - for panel nodes, 235
  - retrieving settings, 97
  - row style template file, 161–162
  - rows made up of, 103
  - sharing, 36–38
  - for simple view, 129–130
  - template, 167–174
  - using widgets, 29, 43–47
  - View UI settings for, 122
  - visual media, 55–56
  - what to know before creating, 17
- Fields tab, content types page, 22**
- File attachments, panel nodes, 235**
- File field type, 56**
- File views, 115**
- FileField module, 55–56**
- files, adding to themes, 68**
- filter handlers, Views API, 316**
- filtered text, 40, 46**
- filters**
- arguments used as, 138
  - building SQL queries in Views, 107
  - creating simple view, 130
  - for criteria in Views list page, 110–111
  - exposed, 145–147
  - relational database, 105–106
  - sorting panel page, 228
  - Views UI, 123–124
- Flag module, 305**

**Flexible layouts, 275–278****Flexinode module, 8–9****Float field type**

- available widget types, 46
- defined, 26
- working with, 40

**Force single setting, exposed filters, 146–147****foreign keys, 104–105****form operation, hook\_field\_settings( ), 80****formatter modules, 87–91****formatter\_info( ), 88–89****formatters**

- CCK fields composed of, 76
- changing how field is displayed, 24
- creating field instances, 91
- creating new, 76–77
- creating themes, 61
- customizing, 77–79
- formatting content, 97

**Forum topic content type, 5****forums**

- getting help with Drupal, 311
- on status of CCK, 9

**Full node menu, Display fields, 35****full node view, 6****Full pager setting, Views UI, 120, 135****functions, theme, 59**


---

## G

**Garland, 59–61****General setting, Variants tab of Panel UI, 230****Getting Started page, Views list, 110****global null variable, arguments, 137**

**global settings**

- Computed Code field, 53
- Date module, 49–50
- fields across all node types, 43–45

**grab handles, moving field into content type, 25–26****Granularity setting, Date, 49–50****grid style, views, 114, 160****GROUP BY clause, query object, 199****grouping in template**

- grouping fields, 170–174
- overview of, 169–170
- second method for, 174–175

**groups**

- adding relationships in View, 122
- changing field display in, 33–34
- creating computed value, 52
- creating with User Reference, 42–43
- field management, 25

**guidelines, module, 309–310**

---

## H

---

**handbooks**

- reading Drupal documentation in, 311
- writing pages for, 314

**handlers**

- naming conventions, 193–194
- plugins vs., 210–212
- properties and methods for query object, 195–199
- properties and methods for views\_object, 194–195
- utilized in Views, 193, 315–317
- Views classes for, 318–324

**headers, adding to views, 120–121****helper functions, for custom programming, 97–99****helper modules**

- add-on, 14
- defined, 13
- Panels, 222
- for site deployment, 299
- for themes, 69–73
- types of, 14

**Hidden option, Display Fields tab, 68****hook\_content\_build\_modes( ), 98****hook\_content\_generate(), 84****hook\_content\_is\_empty(), 84****hook\_content\_notify( ), 78****hook\_field( ), 82–83****hook\_field\_info(), 79–80****hook\_field\_settings(), 80–82****hook\_form validation and submission hooks, 95****hook\_form\_alter( ), 77****hook\_formatter\_info( ), 88–89****hook\_theme( ), 90–91****hook\_views\_admin\_links\_alter hook, 205****hook\_views\_data( ) method, 206–209****hook\_views\_handlers, views\_object, 194****hook\_views\_plugins, views\_object, 194****hook\_views\_pre\_build hook, 202–203, 205****hook\_views\_pre\_execute hook, 203, 205****hook\_views\_pre\_render hook, 203, 205****hook\_views\_pre\_view hook, 205****hook\_views\_query\_alter hook, 205****hook\_widget( ), 87****hook\_widget\_info( ), 85–86****hook\_widget\_settings( ), 86–87**

**hooks**

- CCK's API using, 75
- creating custom field types, widget types or formatters, 77–78
- database schema and, 206–210
- defined, 76
- execution-related, 204–205
- field type module. *See* field type modules
- finding source of, 76
- formatter module, 88–91
- .install file for modules, 78–79
- overview of, 200
- Views implementing, 200–201
- widget type module, 85–87
- hooks\_views\_handlers( ), **registering handlers, 211**
- hooks\_views\_plugins( ), **registering plugins, 212–213**
- HTML files, 156

## I

- icons, styling views, 114
- Identification, content types page, 18–19
- Image field type, Email, 55
- Image modules, 55–56
- ImageAPI module, 56
- ImageCache module, 56
- ImageCache UI module, 56
- ImageField module, 56
- implicit relationships, 191, 206
- importing
  - content types page Import tab, 22
  - Import/Export tool, 281
  - Panels UI Import variant option, 230
  - variants for taxonomy override, 251
  - Views Import module for, 306
  - Views UI Import page, 125

**indexing**

- query optimization with, 182
- relational databases, 103
- \$info array, **CRUD API, 91–92**
- .info file, 61
- informational messages, **Flexible layouts, 275**
- inheritance, in object-oriented programming, 189–190
- init( ), **view execution cycle, 201**
- initialization, **view execution cycle, 201–203**
- inner joins, 106–107
- Input format, **node content, 7**
- insert operation, hook\_field( ), 82
- .install file, 78–79
- Integer field type
  - available widget types, 46
  - defined, 26, 40
- IRC, **getting help with Drupal, 311**
- issue queues, 311, 314
- Items per page option, **Views, 120**

## J

- JavaScript, **Basic tab settings for, 127**
- JOIN statement
  - query object, 196
  - sharing fields between content types, 36
- joins
  - base tables and relationships, 191–192
  - complex, 208–209
  - relational database, 106–107
- JonBob (Jonathan Chaffer), 9
- jQuery UI, **for Panels in-place editor, 222**

---

## K

**keys**

- registering handlers, 211
- relational database, 103–105

---

## L

**Label field**

- changing field display, 33–34
- creating field with complex settings, 31
- field management, 25

**layout**

- changing, 279–280
- node override, 249
- using Flexible, 275–278
- using Panels for. *See* Panels
- using templates for, 156
- Views Bonus Pack pre-created panel, 306

**layout designer, 279****layout manager, 276–277, 279–280****Layout setting, Variants tab of Panel UI, 231****“learning cliff,” Drupal, 102****left joins, 106–107****left-side tabs, Views, 115–117****life cycle, bugs, 312–313****life cycle, Views, 201–203****Limit list to selected items, exposed filters, 146****Link module, 54–55****Link to view, Pane settings in Views, 268****links**

- creating reverse, 43
- in full node view, 6
- in new content type, 12
- in title bar on Views list page, 112

**List page**

- content types, 18
- Panel pages, 228
- Views, 110–112

**list style, Views, 114, 160****live preview, Views UI**

- Basic tab settings, 126–127
- creating simple view, 130
- overview of, 125
- user feeds with arguments, 144

**load operation, hook\_field( ), 81****local tasks (tabs), full node view, 6****locked value, field instances with  
CRUD API, 93****locks, overriding core display pages, 247**


---

## M

**machine names, 24****mailto:link option, Email module, 55****maintenance, and embedded queries, 180****Manage fields, content type List  
page, 18****Manage Fields tab**

- adding fields, 24–25
- adding form elements to, 98
- changing field display in, 33–36
- creating field for existing content type, 27–29

**Maximum length, creating field  
with, 31–32****Maximum setting, widget types, 46****media, creating highly visible, 55–56****Menu module, 4****Menu settings**

- Pages, 121
- panel nodes, 235
- Panels UI, 230

**methods**

- object-oriented programming, 186–187
- query object, 194–195
- views\_object, 194–195

**Mini panels tab, Panels dashboard, 227****Minimum number of words field, Submission forms, 20–21****Minimum setting, widget types, 46****Mini-pager setting, Views UI, 120****mini-panels**

- content editing, 238
- context editing, 258–259
- creating, 220, 237
- creating from Panels dashboard, 226
- editing, 237
- overview of, 236–237
- Panels package, 222

**Module name box, Bulk export page, 127****modules**

- Advanced Profile Kit for Panels, 308
- ApacheSolr, 306
- Calendar, 308
- CCK. *See* CCK (Content Construction Kit) module
- Computed Field, 51–54
- CTools. *See* CTools (Chaos Tool Suite) module
- Date. *See* Date module
- Email, 55
- field type. *See* field type modules
- Flag, 305
- formatter, 87–91
- helper. *See* helper modules
- Image, 55–56
- importance of nodes in, 4
- Link, 54–55
- Menu, 4

## Nodequeue, 304–305

Panels. *See* Panels module

## Pathauto, 303–304

reporting issues about. *See* reporting issues

## Sheetnode, 307

## SimpleViews, 307

## themes directory in, 59

## Total Control Admin Dashboard for Panels, 308

## types used primarily in this book, 10–11

## Upload, 4

Views. *See* Views module

## Views Attach, 306

## Views Bonus pack, 305–306

## Views Bulk Operations (VBO), 307

## Views Datasource, 307

## Views Import, 306

## Views Slideshow, 305

## Views\_or, 304

## widget type, 85–87

**More link, 120, 163****multiple-value formatters, 88–90****multi-row text area widget, 45, 46****MySQL, as install-ready database, 101**


---

## N

**Name field**

- changing field display, 33–34
- field management, 25
- Identification, 18–19

**naming conventions**

- field templates, 62–63
- fields, 79
- handlers, 193–194, 212
- hooks, 200

- indexes, 103
  - plugins, 193–194, 213
  - Views classes, 154
  - Views templates, 157–158
  - widgets, 79
  - New content behavior, Panel, 227**
  - New field setting**
    - changing display of, 34
    - with complex settings, 30
    - for existing content type, 27
    - management, 25
  - New group**
    - changing field display, 34
    - field management, 25
  - new keyword, instantiating constructors, 190**
  - news sites, using taxonomy terms, 262**
  - node, 7**
  - Node add form context, 262**
  - node author relationship, 272**
  - Node content, node override, 249**
  - node contexts**
    - adding, 260–262
    - defined, 258
    - in Panel pages, 258–259
  - Node data type, 26**
  - Node Reference field type**
    - available widget types, 46
    - choosing, 39
    - creating, 41–42
    - defined, 26
    - relationships and, 272–273
    - working with, 41
  - Node Revision views, 115**
  - node templates, creating themes, 65–66**
  - node view, 115, 248–250**
  - node\_add\_form context, 258**
  - node\_edit\_form context, 258**
  - node\_load( ), 95–97**
  - node\_revisions table, 7, 191**
  - node\_save( ), 95–97**
  - node\_submit( ), 97**
  - node/NID, 5–6**
  - Nodequeue module, 304–305**
  - Nodereference module, 14**
  - nodes**
    - adding as content, 239
    - adding fields to, 7
    - creating data for CCK fields, 95–97
    - default content types, 4–5
    - importance of, 4
    - overview of, 3
    - parts of, 5–7
    - settings for fields, 43–44
    - template for, 164
    - turning user profiles into, 272
  - node.tpl.php, 65–66**
  - nofollow option, Link module, 54**
  - Number fields, 40, 76**
  - Number module, 14**
  - Number of values setting**
    - Date module, 49–50
    - field for existing content type, 29
    - field with complex settings, 31–32
  - numbers**
    - field types for storing, 40–41
    - sorting string fields with, 106
  - numeric data types, 103, 106**
- 
- O**
- 
- object-oriented architecture, in Views, 185**
  - object-oriented programming, 186–190**
  - objects, in Views, 192–193**

**online references**

- add-on modules, 14
- CTools, 222
- EXPLAIN statement, 181–182
- getting help with Drupal, 311
- installing Drupal, 10–11
- Organic Groups, 42
- Panels package download, 222
- source control software, 292
- Views modules, 304–307

`option_definition()`,  
`views_object`, **194**

**Optional setting, exposed filters, 146**

`$options` array, `views_object`, **194**

**ORDER BY statement, SQL queries in Views, 107**

**Organic Groups, 42**

**Override setting, Views UI, 124, 134**

**overriding**

- core node editing pages, 253–255
- handlers, 210
- in object-oriented programming, 189–190

**overriding core display pages**

- node view, 248–250
- overview of, 246–247
- taxonomy, 251–252
- user view, 252–253

---

## P

**Page caching, anonymous, 182**

**Page content type, 4**

**Page display**

- creating, 133–135
- creating template with default message, 168
- defined, 116
- display-specific settings for, 121

- grouping fields in template, 170
- user feeds with arguments, 143–144

**Page manager, CTSuite, 222**

**Pager settings, Views UI, 120**

**Pane settings, in Views, 268–269**

**Panel content screen settings, 238, 250, 252**

**panel nodes**

- content editing screen, 238
- creating, 233–236
- creating from Panels dashboard, 226
- editing, 236
- overriding core editing pages, 253–255
- overview of, 233
- Panels package, 222
- tab on Panels dashboard, 227

**panel pages**

- content editing screen, 238
- context editing, 258–259
- creating, 231–233
- creating from Panels dashboard, 226–227
- overview of, 227–229
- user interface, 229–231

**panel panes, adding views to, 266–267**

**panel regions**

- applying styles, 283–284
- Panel content screen settings, 238
- point-and-click layout in Panels, 219–220

**panels**

- access rules for, 244–246
- adding content, 237–242
- caching, 242
- configuring existing content panes, 243–244
- mini-panels, 236–237

- overriding core display pages, 246–253
- overriding core node editing pages, 253–255
- panel nodes, 233–236
- panel pages. *See* panel pages
- Panels dashboard, 225–227
- your first panel, 225
- Panels dashboard, 226–227**
- Panels in-place editor, 222–223**
- Panels module**
  - arguments in panes, 264–271
  - brief history of, 217–218
  - Chaos Tool Suite (CTools), 223
  - contexts. *See* contexts
  - creating contexts, 221
  - exporting, 298–299
  - as modules, 222–223
  - Panels package, 222
  - pluggable architecture, 221
  - point-and-click layout, 219–221
  - purpose of, 217
  - push and pull, 218–219
  - relationships, 271–273
  - theming. *See* theming Panels
- `panels_admin` **stylesheet, 285**
- `panels_dashboard` **stylesheet, 285, 286**
- `panels_dnd` **stylesheet, 285**
- `panels_page` **stylesheet, 286**
- parent directive, plugins, 213**
- patches, writing, 313–314**
- Path, Views list page, 112**
- path directive, plugins, 213**
- Path module, 4**
- path setting, Page, 121**
- Pathauto module, 308–309**
- paths, node, 5**
- patience, reporting issues and, 313**

## performance

- Basic tab settings, 126–127
- caching hooks to improve, 201
- determining query, 178–179
- embedding queries for, 179–180
- sharing fields between content types, 36
- using EXPLAIN for query, 180–182

## permissions

- to access published content, 6
- changing user roles or, 120
- Content Permissions module, 14
- managing access rules, 244–246
- node view, 248
- overriding core display pages, 247
- overriding core node editing pages, 254
- Panels UI settings, 230–231
- User Reference vs. Node Reference, 43
- view pane displays, 269

## PHP

- CCK's API using, 75
- Computed Field module requiring, 51–54
- creating relative dates with, 48–49
- embedding queries, 179–180
- templates and, 156

## plain text

- Text field type, 40
- widget for Email module, 55
- widget options for Text field type, 46

## players, adding with User Reference, 42–43

## plugins

- handlers vs., 210–212
- naming conventions, 193–194
- Panels architecture, 221
- Panels styles, 290



**plugins** *(continued)*

- properties and methods for query object, 195–199
- properties and methods for views\_object, 194–195
- Views, 192–193
- Views API, 317–318
- Views classes for, 324–326
- podcasts, getting help using, 311
- point-and-click layout, Panels, 219–221
- Poll content type, 5
- polymorphism, in object-oriented programming, 189
- possible\_keys, query optimization, 181
- Post date, node content, 6
- PostgreSQL, 101
- posts, nodes as, 3
- pre\_execute( ), view execution cycle, 202
- pre\_query( ), view execution cycle, 202
- pre\_render( ), view execution cycle, 203
- <pre> tag, HTML, 165
- Precision setting, widget types, 46
- Prefix setting, widget types, 46
- prepare translation, hook\_field( ), 82
- preprocessors, for themes, 164
- presave operation, hook\_field( ), 82
- Preview, node override, 250
- preview( ), executing views display, 204
- Preview ,Variants tab of Panel UI, 231
- primary keys, 103–105
- Print page, Display fields, 35–36
- printing, default messages for empty fields, 167–169
- procedural architecture, 185–188
- production, moving from development to, 293–294

**profiles**

- creating specific user, 308
- turning into nodes, 272

**Promoted to front page option**

- node content, 6
- Workflow, 21

**properties**

- field, 24
- query object, 194–195
- views\_object, 194–195

**Published filter, 124****Published option**

- node content, 6
- panel nodes, 236
- Workflow settings, 21

**pulling in needed content, Panel layout, 219****pushing in needed content, Drupal layout, 218**


---

## Q

**queries, SQL**

- Basic tab settings, 126–127
- constraining with primary keys, 103–105
- filtering and sorting, 105–106
- joins and, 106–107
- in Views, 102, 107–108

**query( ) method, query object, 196****query object**

- properties and methods, 195–199
- view execution cycle, 203

**query optimization**

- determining query performance, 178–179
- development time vs. CPU time in, 177
- embedding queries for, 179–180

- experimenting with, 183
- EXPLAIN statement, 180–182
- indexing vs. caching, 182
- not worrying about, 177–178
- overview of, 177

## R

---

**radio button widget, 45–46**

**records of data. See rows**

**Reference data type, 26**

**Region menu, Flexible panel layout, 277**

**registering**

- handlers, 211
- plugins, 212

**registry, rebuilding theme, 68**

**relational databases**

- emergence of Views for, 102
- filtering, 105–106
- joins, 106–107
- keys, 103–105
- rows and fields, 103
- sorting, 106
- SQL and, 101
- from SQL to Views to human language, 107–108

**relationships**

- adding, 122
- base tables and, 191–192
- building SQL queries, 107
- creating display using, 150–151
- designing page content and layouts using, 280
- with foreign keys, 104–105
- handlers for, 317
- joins and, 106–107
- overview of, 148–149
- in Panels, 271–273

- query object, 196–198

- relating tables to each other, 206–207
- in taxonomy override page, 251

**relative dates, creating, 48–49**

**release notes, reporting issues, 311**

**Remember setting, exposed filters, 146**

**render( ) method, view execution cycle, 203**

**reporting issues**

- asking for another's time and, 314
- bug requests vs. support requests, 312
- checking other sources, 311–312
- contribute back, 314
- life cycle of bugs, 312–313
- overview of, 309
- patience, 313
- reading documentation, 310–311
- staying on topic, 312
- submitting complete report, 309–310

**reports, submitting complete, 309–310**

**Required setting**

- creating field for existing content type, 29
- creating field with complex settings, 31–32
- Date module, 49–50

**Rescan template files**

- creating template with default message, 168
- finding new template files with, 156
- grouping fields in template, 171
- overview of, 165–166

**reverse links, creating, 43**

**revision control, 292**

**revision ID (tid) primary key, 103–104**

**Revision information settings, panel nodes, 235**

**river of news listings, 6**

**roles**

- creating for views, 120
- User Reference and limitations on user, 43

**Row menu, Flexible panel layout, 277**

**Row style plugin, 212**

**row styles, Views**

- defined, 114–115
- overview of, 193
- template files, 161–162

**rows**

- in relational database tables, 103
- reliance on keys, 103–105
- simple views with, 129

**RSS feeds**

- with arguments, 143–144
- creating feed view, 135
- display template for, 159
- row style template file, 162

**RSS page, Display fields, 34–36**

---

## S

**sanitize operation, hook\_field( ), 82**

**Save content type button, 22**

**save operation, hook\_field\_settings( ), 80**

**saving views, 142, 144**

**screencasts, for help with Drupal, 311**

**select list widget, 45–46**

**SELECT statement, SQL queries, 107**

**selection rules, 230, 247**

**Send arguments, Pane setting in Views, 269**

**Separator field, table style view, 114**

**set\_arguments( ), view execution cycle, 201**

**set\_where\_group( ), query object, 198**

**Settings tab, Panels dashboard, 226–227**

**Settings tab, Panels UI, 230**

**sharing, fields between content types, 29, 36–38**

**Sheetnode module, 307**

**SimpleViews, 307**

**single on/off check box widget, 45–46**

**single-value formatters, 88–89**

**site deployment**

- configuring development environment, 291–293
- exporting CCK, 295–296
- exporting panels, 298–299
- exporting views, 296–297
- exporting your structures, 294–295
- helper modules for themes, 299
- moving to production, 293–294

**/sites/all/themes directory, base theme, 60–61**

**slider bar, Flexible panel layout, 277–289**

**slideshows, creating, 305**

**Sort Criteria settings**

- creating simple view, 130
- defined, 123
- user feeds with arguments, 143

**sort handlers, Views API, 315–316**

**sorting**

- criteria applied in Views list page, 111
- relational database, 106
- tables by columns, 114

**source code, CSS in, 285–289**

**source control, configuring development environment, 292–293**

**special characters, Name and Type fields, 19**

**spreadsheets, creating, 307**

**SQL (Structured Query Language)**

- emergence of Views, 102

- filtering and sorting, 105–106
    - joins, 106–107
    - relational database basics, 102–105
    - relational databases and, 101
    - using CCK's API, 75
    - to Views to human language, 107–108
  - stacking arguments, 144–145**
  - stay on topic, in reporting issues, 312**
  - Sticky, node content, 6**
  - Sticky at top of lists, Workflow settings, 21**
  - storage, database, 8–9**
  - Story content type, 4–5**
  - string context, 258**
  - string data types, 103, 106**
  - strtotime function, PHP, 48–49**
  - structures**
    - changing when sharing fields between content types, 36
    - configuring development environment, 291
    - exporting. *See* exporting structures
  - style plugins**
    - defined, 212
    - Panels, 290
    - Views, 193, 317–318
  - styles, Panel**
    - applying, 283–284
    - creating new, 282–283
    - CSS in, 284–285
    - other stylistic changes, 289–290
    - Panels module supporting, 221
    - sharing, 281
    - using Stylizer, 280–281
  - styles, View**
    - Basic settings, 119–120
    - creating simple view, 129
    - feed display, 135
    - icons for setting, 114
    - overview of, 160
    - in themes, 61
  - stylesheets, 285–286**
  - Stylizer**
    - creating new panel style, 282–283
    - defined, 222
    - overview of, 280–281
  - submenu bar, Views Add page, 112**
  - Submission form settings, content types page, 19–21**
  - Suffix setting, widget types, 46**
  - suggestions, overriding default theme files, 62**
  - Summary setting, Variants tab of Panel UI, 230**
  - Summary tab, Panels UI, 230**
  - summary views, templates for, 164**
  - support requests, bug reports vs., 312**
  - system pages, 228**
- 
- ## T
- 
- table style, Views, 114, 160–161**
  - tables**
    - declaring fields on, 209–210
    - declaring in `hook_views_data()`, 207–209
    - defined, 103
    - filtering and sorting fields, 105–106
    - joins, 106–107
    - keys, 103–105
    - relating to each other, 206–207
    - rows and fields, 103
  - tabs**
    - Panels UI, 230
    - Views left-side, 115–116
  - Tag, Views list page title bar, 112**

**tags**

- changing for default view, 113–114
- editing for default view settings, 117–118

**tasks, 312****taxonomy**

- adding term to pane, 263–264
- child vs. sibling terms, 264
- creating list of terms, 115
- overriding core display pages, 251–252
- overriding title, 264
- using arguments for depth of, 137
- using panels with, 262
- using relationships in panels, 272

**teams, creating, 42–43****Teaser**

- Display fields, 35
- node content, 7

**teaser view, node view as, 6****templates, theme**

- changing, 68–69
- field, 61–64
- node, 65–66
- overview of, 59–60
- using Theme Developer, 70–71
- viewing with Contemplate module, 71–73

**templates, Views**

- debugging, 167
- display, 159–160
- grouping, 169–175
- listing default, 156–157
- looking inside, 166–167
- other, 162–164
- printing default messages for empty fields, 167–169
- Rescan template files, 165–166

row, 161–162

view styles, 160–161

working with, 164–165

**Term views, 115**

`term(s)` context, 258

**testing**

- on non-production site, 293–294
- query performance, 178–179

**text, title bar on Views list page, 112****Text field type**

- available widget types, 46
- defined, 26
- working with, 40

**text field widget, 45–46****Text module, 14****Text processing setting, 31–32****Theme Developer, 70–71**

`theme.inc` file, /views/theme directory, 165

**themes**

- adding files to, 68
- basics, 59–60
- changing parts of, 64–65
- excluding fields from, 68
- field templates creating, 61–64
- for fieldgroups, 66–68
- formatters creating, 61
- helper modules for, 69–73
- node templates creating, 65–66
- nodes enabling, 4
- overview of, 59
- preserving base theme, 60–61
- refining master layout, 218
- settings in View, 121
- templates. *See* templates, theme
- using Node Reference, 68–69

**theming Panels**

- changing layouts, 279–280
- CSS classes, 289
- CSS in Panels UI, 284–285
- CSS in source code, 285–286
- Flexible layout, 275–278
- identifying particular pane, 286–288
- other stylistic changes, 289–290
- using Stylizer, 280–284

**theming Views**

- classes, 153–156
- overview of, 153
- template files. *See* templates, Views

`$this` variable, OOP, 186–187

`tid` (revision ID) primary key, 103–104

**Time zone handling field, Date, 49–50**

**title**

- Basic settings for Views UI, 119
- configuring arguments for, 140
- executing views display for page, 203–204
- node content, 6
- Panel content screen settings, 238
- Submission form settings, 19
- Views list page, 112

**To Date setting, Date module, 49–50**

**Tools pages, Views UI, 125–126**

**topic, reporting issues by staying on, 312**

**Total Control Admin Dashboard, 308**

`.tpl.php` extension, themes

- field templates, 62
- node templates, 65–66
- overview of, 59

**tutorials**

- creating page, 254
- getting help with Drupal, 311

**Type field**

- adding fields, 25
- changing field display, 33–34
- Identification, 18–19

`type` setting, CRUD API, 92

`type_name`, CRUD API, 91

---

## U

**UI (user interface), Panels, 229–231**

**UI flow, handlers, 210**

`uid` (user ID) primary key, 103–105

**underscores, in machine names, 24**

**unformatted style, Views, 114, 161**

**Unlock operator, exposed filters, 146**

**updates**

- `hook_field()`, 82
- node content, 6
- node override, 250
- Panel content screen settings, 238

**Upload module, 4**

**URLs**

- creating readable and memorable, 308–309
- node, 5–6
- not passing arguments to blocks with, 144–145
- overriding using Pane settings in Views, 269
- path settings, creating panel nodes, 235
- taxonomy override page and, 251
- using arguments to hide/remove information in, 137

**Use Ajax option, Views UI, 120**

**Use panel path option, Pane settings in Views, 268**

`user` context, 258

**user interface (UI), Panels, 229–231**

**user profiles**

- creating specific, 308
- turning into nodes, 272

**User Reference field type**

- available widget types, 46
- choosing, 39
- defined, 26
- and relationships, 272–273
- working with, 42–43

**user view**

- defined, 115
- overriding core display pages, 252–253

**Userreference module, 14****users, reporting issues. *See* reporting issues**


---

## V

---

**validate operation, hooks, 80, 82****validation**

- of arguments, 140
- Views UI, 124

**validator plugins, Views, 193****variables**

- changing theme with Node Reference, 68–69
- field template, 63–64

**variants**

- node override, 249–250
- taxonomy override, 251–252

**Variants tab, Panels UI, 230****version control, using source control, 293****version numbers, Drupal and, 10****vid (version ID) primary key, 103–104****view building flow, handlers, 210****View Bulk Operations (VBO), 307****view execution cycle, 201****view object, Views, 192****view pane displays, 268–271****view\_pre\_view hook, 202****.view-content class, 154****.view-empty class, 154****.view-filters class, 154****.view-footer class, 154****.view-header class, 154****Views API**

- base tables and relationships, 191–192
- classes, 318–326
- data architecture, 185
- database schema and data hook, 206–210
- executing views display, 203–204
- execution-related hooks, 204–205
- handlers, 210–212, 315–317
- life cycle of view, 201–203
- object-oriented programming, 186–190
- objects involved in a View, 192–199
- overview of, 199–201
- plugins, 212–213, 317–318

**Views Attach module, 306****Views Bonus pack, 297, 305–306****Views content panes, CTSuite, 222, 226****views data operation,  
hook\_field\_settings( ), 81****Views Datasource module, 307****Views Datasource plugins, 307****views directory, 156****Views Import module, 306****Views module**

- Add content modal for, 266–267
- Add creation page, 115–116
- birth of Views 2, 102
- Calendar module requiring, 308
- creating. *See* Views UI

- emergence of, 102
- exporting, 296–297
- exposed filters, 145–147
- filters, 40
- node, 6
- overview of, 107–108
- Pane settings, 268–269
- query optimization. *See* query optimization
- relationships, 148–151
- saving, 142
- settings, 117–118
- simplifying queries with, 107
- templates. *See* templates, Views
- theming. *See* theming Views
- using arguments. *See* arguments
- Views API. *See* Views API

#### **Views modules**

- ApacheSolr, 306
- Flag, 305
- Nodequeue, 304–305
- Sheetnode, 307
- SimpleViews, 306
- View Bulk Operations (VBO), 307
- Views Attach, 306
- Views Bonus Pack, 305–306
- Views Datasource, 307
- Views Import, 306
- Views Slideshow, 305
- Views\_or, 304

#### **Views Slideshow module, 305**

#### **Views UI**

- Add pages, 112
- arguments, 122
- Basic settings, 118–121
- Basic tab, 126–127
- Bulk Export page, 127

- Convert tab, 127–128
- creating feed view, 135
- creating simple view, 128–130
- display specific settings, 121–122
- embedding view within view using attachments, 135
- enabling and changing default view, 113–114
- fields, 122
- filters, 123–124
- Import page, 125
- left-side tabs, 115–117
- List page, 110–112
- live preview, 125
- overrides, 124
- overview of, 131–135
- relationships, 122
- sort criteria, 123
- Tools pages, 125–127
- using Advanced Help module for, 110
- validation, 124
- View settings, 117–118
- Views 2 UI vs. original, 109–110
- Views Add page, 115–116

#### **Views\_bookmark module, 305**

- `views_embed_view( )`, **204**
- `views_get_current_view( )`, **202**
- `views_handler`, **193**
- `views_handler_filter_in_operator handler`, **211**
- `views_object`, **properties and methods for**, **194**

#### **Views\_or module, 304**

- `views_plugin`, **193**
- `views-exposed-form.tpl.php file`, **163**
- `views-more.tpl.php file`, **163**
- `views-ui-*.tpl.php file`, **163**



views-view-fields.tpl.php file,  
**161–162, 166, 170–171**

views-view-field.tpl.php file, **164, 168–169**

views-view-grid.tpl.php file, **160**

views-view-list.tpl.php file, **160**

views-view-row-comment.tpl.php file, **164**

views-view-row-node.tpl.php file, **164**

views-view.rss.tpl file, **159**

views-view-rss.tpl.php file, **162**

views-view-summary.tpl.php file, **164**

views-view-summary-unformatted.tpl.php file, **164**

views-view-table.tpl.php file, **160–161**

views-view.tpl.php file, **158–159**

views-view-unformatted.tpl.php file, **160–161**

**visibility rules. See access rules, Panels UI**

**vocabularies**

- adding taxonomy term to pane, 263–264
- building with taxonomy, 262

vocabulary **context**, **258**

---

## W

**Web search, for help with Drupal, 311**

**WHERE statement**

- building SQL queries in Views, 107

- with EXPLAIN for query optimization, 181–182

- implementing filters in SQL, 105–106

### **widget types**

- changing, 47

- creating custom, 77–79

- creating field instances with, 91

- creating new, 76–77

- modules, 85–87

- and settings per field type, 29, 46

widget\_type **setting**, **92**

### **widgets**

- in CCK at install, 45

- in CCK fields, 76

- constraining data with, 43–45

- creating field with complex settings, 31

- Date module, 48

- Email module, 55

- FileField module, 56

- overview of, 29

- template, 163

### **wildcards**

- as shortcut for arguments, 140

- taxonomy override page and, 251

**Workflow, content types page, 21–22**

---

## X

**XAMPP stacks, installing Drupal, 10**