Mastering Drupal 8 Views

Learn how to build complex displays of content—all without programming.

Views was used on more than 80% of all Drupal 7 sites; now it is part of the Drupal 8 core. While many site builders and site owners are aware of Views, they don’t understand how to take full advantage of its power to create many amazing pages and blocks. If they are using Views, they might build 10 different view displays with different filters without knowing that a contextual filter would require only a single display.

Using our sample company, we’ll take its existing content and grow an ever more complex and powerful website for that company, starting by adapting the administration the user sees and moving on to making complex pages of information for site visitors.

Who this book is written for

This book is for web developers, web designers, and website administrators who use Drupal 8, have some basic knowledge of managing and developing apps with Drupal, and who want to get advanced, practical knowledge of Views and how to leverage it in Drupal 8 applications.

What you will learn from this book

- Create displays of content that are automatically updated when you add new content
- Show maps and rotating image carousels on your site
- Combine content to create composite displays using the same data in different ways
- Use fields from more than one content type to create powerful views of multitable data
- Modify the field data being displayed, combining it in different ways or changing the HTML that might be normally generated
- Add headers and footers above and below a views display
- Make the output of your view look exactly as the designer specifies
- Change the order in which a list of content is displayed, and limit the content shown to certain content types or values
- Filter content using values passed to the view in the URL, making a single view work in different ways

Accelerate your development process using Drupal 8 Views with this advanced, practical guide

Foreword by Jeffrey A. "jam" McGuire - Evangelist, Developer Relations, Acquia
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Up and Running with Views'
- A synopsis of the book’s content
- More information on Mastering Drupal 8 Views
About the Author

Gregg Marshall has been using Drupal since 2006. While he installed Drupal 5, his first production site was Drupal 6. He started his career as a software engineer, switched to sales/marketing in an unrelated industry, and drifted back to development. After attending DrupalCon 2010 in San Francisco, Gregg switched from channel marketing that used Drupal for microsites to being a full-time Drupal developer.

Gregg works as a consultant/contractor, frequently mentoring in-house development teams on Drupal best practices. His current contract is with WebNY, where he serves as the senior Drupal architect for the state of New York, helping build a single platform for all the state's websites.

Gregg is a speaker at Drupal Camps and is a professional speaker to 30+ associations and corporate meetings. He has over 400 published articles in a variety of publications. Gregg has a bachelor's degree in chemistry, a master's degree in electrical engineering, and a master's degree in business administration. He is an Acquia Certified Drupal Grand Master.
To make this book a little easier to read, I use a story of a Drupal user who has advanced to the point of being able to do their own site building. Setting the stage for the story woven into the book, let's start with a bit of background for the story.

Blue Drop Realty started business in 2002 as a part-time "hobby" of the owner, Lynn. As their business grew, they decided to build a website in 2005 using Dreamweaver to lay out their pages. This worked okay for about 5 years, but it was taking too much of Lynn's time to update the pages, and the site's design was starting to look dated. The business had grown enough that they had a new website developed by Fancy Websites, Inc., who used Drupal 6 to build the new website. With Drupal, the office manager and even some of the salespeople were able to make changes to the pages.

Lynn is joined in this story by her grey tabby cat, Jackson, an alley cat, literally, who adopted Lynn's family a couple of years ago. Once Jackson got comfortable with the new surroundings and being around a family, he frequently would curl up on Lynn's desk next to her monitor when she would work at night. Lynn has gotten into the habit of talking to Jackson as she works—he has proven to be a good listener.

Their Drupal 6 website is still working fine for Blue Drop Realty, but knowing that when Drupal 8 is released, support for Drupal 6 will end shortly afterward, Lynn is considering options to upgrade to the latest version of Drupal. She has gotten comfortable with Drupal 6 and even makes some "programming" changes to the site using the Drupal user interface as a site administrator. Lynn was able to add a news release content type herself, but she had to have Fancy Websites, Inc. create the views to display a list of releases on a news page. As she worked with Drupal, Lynn learned how to modify some views and even build a relatively simple list view on her own.
Jim is the developer at Fancy Websites, Inc., which built Lynn's original Drupal 6 site. As Lynn learns and becomes more comfortable with Drupal, Jim will morph from the developer who built the site to Lynn's mentor, as she herself will grow from a new content editor to site builder. More than anything else, Jim will help Lynn whenever she gets stuck.

A short history

Drupal is an open source content management system used by over 1,200,000 websites. It started as a message board written by Dries Buytaert in 2000 while in college at University of Antwerp. The original site Dries built was drop.org, a mistake when he went to register dorp.org, dorp being Dutch for village. When he decided to open source the software in January 2000, he named it Drupal.

Views has been described as the report writer for Drupal. It is that and a whole lot more.

Drupal progressed rapidly through many revisions, with Drupal 4 released in June 2002, Drupal 5 in January 2007, Drupal 6 in February 2008, and Drupal 7 in January 2010. Drupal 8, which is a major rearchitecture of Drupal, has taken almost 5 years, a third of Drupal's total lifetime, to be completed.

In May 2006, a contributed CCK module, Content Construction Kit, was released for Drupal 4.7 and became the standard for defining fields via the user interface, replacing the older Flexinode module, which dates back to February 2004. Before this, any modifications to the basic structure of a node would require manually defining database tables for a field and creating all the functions to create, edit, display, and delete the field when the corresponding node is changed or viewed.

The Views module was first created (at least the first commit was made to it) on November 25, 2005. The first release, 4.6.x-1.x-dev, was on December 1, 2005. The first non-development release, 4.7.x-1.0, was almost a year later on November 11, 2006. The first release contained 3,177 executable lines of PHP. The current Drupal 7 release of Views contains 57,155 lines of non-comment PHP. Its supporting module, CTools, contains another 39,939 lines. The Drupal 8 version of Views contains 58,925 lines, with most of CTools absorbed into Drupal core. Just as telling is that Drupal 8 Views has 19,921 lines of tests, where Drupal 7 only has 9,238 and Drupal 4.7 has none.

Before the Views module, any display not contained in the Drupal core or a contributed module required manually creating the necessary SQL queries, executing them against MySQL (at the time the only database supported by Drupal), and then taking the results and formatting them into HTML. This required significant PHP and MySQL programming experience. With the release of Views, users with limited programming experience could create powerful displays from the user interface.
At the time of writing, Views for Drupal 7 has almost 600 supporting contributed modules. As of the release of Drupal 8, Views for Drupal 8 has 65 supporting contributed modules, but many module contributors are just starting to port modules to Drupal 8, so the number will rise dramatically in the months to come.

Drupal is always changing
This book was started just as Drupal 8.0.0 was about to be released and was finished shortly before Drupal 8.1.0 was about to be released. In between were 8.0.1 through 8.0.5, a new release about every two weeks. With each release, a number of issues were resolved. Some of those issues affect how Views operates or the wording of some screen text or prompts. The evolution of Drupal and Views will continue long after this book is done. As a result, it is very likely that some screens and prompts described here might not match your copy of Drupal exactly. The differences are minor changes in wording to improve the understandability or clarify the meaning. Do not be alarmed if there are small differences between this book and Views when you use it. These differences will be mostly cosmetic, although it is possible that a new option might appear.

What this book covers
The first thing you will notice is that this isn't your traditional technical book written in a dry third-party tone. It reads a bit more like a novel, following Lynn as she learns how to master Views. Jackson, her cat, and Jim, her mentor, are along for the ride and to let the story change viewpoints from time to time (and to give Lynn "people" to talk to).

Views is an amazing part of Drupal 8. I hope you will enjoy this approach to explaining it.

Chapter 1, Up and Running with Views, deals with the Views main administration page and setting pages and does a quick run-through of View's edit screen by modifying an existing core view.

Chapter 2, Views from Scratch, starts the in-depth exploration of Views, starting with building a simple property listing that most real estate sites might have. We will define some Views terms such as View Types, displays, Display Formats, and Display Contents. We'll end with an interesting view that displays selected events from Drupal's watchdog log to users that might normally not be able to see them.
Chapter 3, Sorting and Filtering, introduces sorting and filtering. Sorting lets you specify the order you want your results to be displayed in. We will move on to limiting which results are even displayed using filtering. You can also expose a filter to the user and let them select which results to show. Finally, we'll show how to group the results when a view results in multiple rows for each value.

Chapter 4, Contextual Filters, takes filtering to the next level by adding contextual filters. Contextual filters use the URL to pass the selection criteria to the filter.

Chapter 5, Relationships, shows how to use relationships to gain access to even more data to display. Using the built-in entity reference, any reference field can be used to add all the fields in the referenced content to the available fields list.

Chapter 6, Add-on Modules, adds more Display Formats using add-on contributed modules. Using add-on modules, we can add a slideshow or a rotating carousel as easily as enabling the module, installing the JavaScript library it uses, and defining a very simple view.

Chapter 7, Field Rewrites, is about field rewrites. It is uncommon that the default output of a view exactly meets your needs, including the if-then-else values, in which if a condition is true, one value is displayed, but if it isn't true, another value is displayed.

Chapter 8, Customizing Views, covers the rest of the options in the center column of the view edit page, such as custom headers, footers, and special messages if a set of filters (defined, exposed, or contextual) results in no results.

Chapter 9, Advanced View Settings, deals with all of the advanced settings except contextual filters and relationships, which are covered in earlier chapters.

Chapter 10, Theming Views, winds up the book with how to theme or style the output of Views. Some CSS support can be easily added through the user interface; other, more complex changes might require replacing the Twig templates used by Views with custom templates that do manipulations not possible from the user interface.
Up and Running with Views

Drupal 8 was released on November 19, 2015, after almost 5 years of development by more than 3,000 members of the Drupal community. Drupal 8 is the largest refactoring in the project's history.

One of the most important changes in Drupal 8 was the inclusion of the most popular contributed module, Views. Similar to including CCK in Drupal 7, adding Views to Drupal 8 influenced how Drupal operates, as many of the administration pages, such as the content list page, are now views that can be modified or extended by site builders.

Every site builder needs to master the Views module to really take advantage of Drupal’s content structuring capabilities by giving site builders the ability to create lists of content formatted in many different ways. A single piece of content can be used for different displays, and all the content in each view is dynamically created when a visitor comes to a page. It is so important, it was the only contributed module included in the Acquia Site Builder certification examination for Drupal 7.

In this chapter, we will discuss the following topics:

- Looking at the Views administration page
- Reviewing the general Views module settings
- Modifying one of the views from Drupal core to create a specialized administrative page
Drupal is always changing

This book was started just after Drupal 8.0.0 was released and has been finished shortly before Drupal 8.1.0 is about to be released. In between are 8.0.1 through 8.0.5, a new release about every two weeks. With each release, a number of issues are resolved. Some of those issues affect how Views operate or the wording of some screen text or prompts. The evolution of Drupal and Views will continue long after this book is done. As a result, it is very likely that some screens and prompts described here might not match your copy of Drupal exactly. The differences are minor changes in wording to improve the understandability or clarify the meaning. 

Do not be alarmed if there are small differences between this book and Views when you use it. These differences will be mostly cosmetic, although it is possible a new option might appear.

Drupal 8 is here, should I upgrade?

"Jim, this is Lynn, how are things at Fancy Websites?"

"Hi Lynn, things are going well."

Lynn continued, "I read that Drupal 8 is being released on November 19. From our conversations this year, I guess that means it is time to upgrade our current Drupal 6 site. Should I upgrade to Drupal 7 or Drupal 8?"

"Lynn, we're really excited that Drupal 8 is finally ready. It is a game changer, and I can name 10 reasons why Drupal 8 is the way to go:"

• Mobile device compatibility is built into Drupal 8's DNA. Analytics show that 32% of your site traffic is coming from buyers using phones, and that's up from only 19% compared to last year.

• Multilingual is baked in and really works, so we can go ahead and add the Spanish version of the site we have been talking about.

• There's a new theme engine called Twig that will make styling the new site much easier. It's time to update the look of your site; it's looking pretty outdated compared to the competition.

• Web services are built in. When you're ready to add an app for your customer's phones, Drupal 8 will be ready.
• There are lots of new field types built in for useful stuff such as telephone numbers, e-mail addresses, and more, so we won't need to add half a dozen contributed modules to let you build your content types.

• Drupal 8 is built using industry standards. This was a huge change you won't see, but it means that our shop will be able to recruit new developers more easily.

• The configuration is now stored in code. Finally, we'll have a way for you to develop on your local computer and move your changes to staging and then to production without having to rebuild content types and views manually over and over.

• The WYSIWYG editor is built in. The complex setup we went through to get the right buttons and make the output work won't be necessary in Drupal 8.

• There's a nice tour capability built in so that you can set up custom "how to" demonstrations for your new users. This should free up a lot of your time, which is good given how you are growing.

• I've saved the best for last. Your favorite module, Views, is now built into core! Between Fields in Drupal 7 and now Views in Drupal 8, you've got the tools to extend your site built right into core.

"The bottom line is I can't imagine not going ahead and upgrading to Drupal 8. Views in core is reason enough. Why don't I set up a Drupal 8 installation on your development server so that you can start playing with Drupal 8? We're not doing any development work on your site right now, and we still have staging to test any updates."

"That sounds great, Jim! Let me know when I can log in."

Less than an hour later, the e-mail arrived; the Drupal 8 development site was set up and ready for Lynn to start experimenting.

Based on the existing Drupal 6 site, Lynn set up four content types with the same fields she had on the current site. These content types are outlined in Appendix, Content Types for a Sample Site. Jim was able to use the built-in migrate module to move some of her data to the new site.

Lynn was ready to start exploring Views in Drupal 8.
Looking at the Views administration page

That evening, Lynn logged into the new site. Clicking on the Manage menu item, she then clicked on the Structure submenu item, and at the bottom of the list displayed on the Structure page, she clicked on the Views option.

About that time, Jackson came in and settled into his spot near her terminal.
"Hi Jackson, ready to explore Views with me?"

Looking at the Views administration page, Lynn noticed there were already a number of Views defined. Scanning the list, she said "Look Jackson, Drupal 8 uses Views for administration pages. This means we can customize them to fit our way of doing things. I like Drupal 8 already!". Jackson purred. Lynn studied the Views administration page shown here:

As Lynn looked at each view, the listing looked familiar; she had seen the same kind of listing on her Drupal 6 site. Trying the OPERATIONS pull-down menu on the first view, she saw that the options were Edit, Duplicate, Disable, and Delete. "That's pretty clear; I guess Duplicate is the same as Clone on my old version of Views. I can change a view, create a new one using this one as a template, make it temporarily unavailable, or wipe it completely off the face of the earth."
"I wonder what kind of settings there are on the Settings tab of this listing page. Look, Jackson, there's a couple of subtabs hiding on the Settings page." As Lynn didn't want to mess up her new Drupal site, she called Jim. "Hi, Jim. Can you give me a quick rundown on the Views Settings tab?"

"Sure," he replied.

**Views settings**

"Looking at the Views Settings tab, you'll notice two subtabs, Basic and Advanced. Select the advanced settings tab by clicking on Advanced to show the following display:

![Advanced Views settings](image)

The Views advanced settings configuration page

**Views advanced settings**

Let's look at the Advanced tab first since you'll probably never use these settings.

The first option, Disable views data caching, shouldn't be checked unless you are having issues with views not updating when the data changes. Even then, you should probably disable caching on a per-view basis using the caching setting in the view's edit page in the third column, labeled Advanced, near the bottom of the column. Disabling views' data caching can really slow down the page loads on your site.

You might actually use the Advanced settings tab if you need to clear all the Views' caches, which you would do by clicking on the Clear Views' cache button.
The other advanced setting is DEBUGGING with an Add Views signature to all SQL queries checkbox. Unless you are using MySQL's logs to debug queries, which only an advanced developer would do, you aren't going to want this overhead added to Views queries, so just leave it unselected.

**Views basic settings**

Moving to the Basic tab, there are a number of settings that might be handy, and I'd recommend changing the default settings. Click on Basic to show the following display:

The Views basic settings configuration page

The first option, *Always show the master (default) display*, might or might not be useful. If you create a new view and don't select either create a page or create a block (or provide a REST export if this module is enabled), then a default view display is created called master. If you select either option or both, then page and/or block view displays are created; generally you won't see master. It's there; it's just hidden.
Sometimes, it is handy to be able to edit or use the master display. While I don’t like creating a lot of displays in each view, sometimes I do create two or three if the content being displayed is very similar. An obvious example is when you want to display the same blog listing as either a page or in a block on other pages. The same teaser information is displayed, just in different ways. So, having the two displays in the same view makes sense.

Just make sure when you customize each display that any changes you make are set to only apply to the current display and not all displays. Otherwise, you might make changes you hadn’t planned on in the other displays. Most of the time, you will see a pull-down menu that defaults to All displays, but you can select This page (override) to have the setting change apply only to this display.

Using the master display lets you create information that will be the same in all the displays you are creating; then, you can create and customize the different displays. Using our blog example, you may create a master display that has a basic list of titles, with the titles linking to the full blog post. Then, you can create a blog display page, and using the This page (override) option, you can add summaries, add more links, and set the results to 10 per page. Using the master display, you can go back and add a display block that shows only the last five blog posts without any pager, again applying each setting only to the block display. You might then go back to the master display and create a second block that uses the tags to select five blog posts that are related, again making sure that the changes are applied to the current block and not all displays.

Finally, when you want to change something that will affect all the displays, make the change on the master display, and this time, use the All displays option to make sure the other displays are updated. In our blog example, you might decide to change the CSS class used to display the titles to apply formatting from the theme; you probably want this to look the same in every possible display of the blog posts.

CSS and theming is covered in Chapter 10, Theming Views.

The next basic setting for Views is Allow embedded displays. You will not enable this option; it is for developers who will use Views-generated content in their custom code. However, if you see it enabled, don’t disable it; doing this would likely break something on your site using this feature.
The last setting before the **LIVE PREVIEW SETTINGS** fieldset is **Label for "Any" value on non-required single-select exposed filters**, which lets you pick either `<Any>` or `-Any-` as the format for exposed filters that would allow a user to ignore the filter.

**Live Preview Settings**

There are several **LIVE PREVIEW SETTINGS** fieldsets I like to enable because they make debugging your views easier. If the **LIVE PREVIEW SETTINGS** fieldset is closed (if these options are not showing), click on the title next to the arrow, and it will open. It will look similar to this:

```plaintext
LIVE PREVIEW SETTINGS

- Automatically update preview on changes
- Show information and statistics about the view during live preview

Show SQL query
- Above the preview
- Below the preview
- Show the SQL query
- Show performance statistics
- Show other queries run during render during live preview

Drupal has the potential to run many queries while a view is being rendered. Checking this box will display every query run during view render as part of the live preview.
```

I generally enable the **Automatically update preview on changes** option. This way, any change I make to the view when I edit it shows the results that would occur after each change. Seeing things change right away shows me whether a change will have an effect I'm not expecting.

A lot of Views options can be tricky to understand, so a bit of trial and error is often required. Hence, expect to make a change and not see what you expect; just change the setting back, rethink the problem, and try again. Almost always, you'll get the answer eventually. If you have a view that is really complex and very slow, you can always disable the live preview while you edit the view by selecting the **Auto preview** option in the gray **Preview** bar just under all the view's settings.
The next two options control whether Views will display the SQL query generated by the Views options you selected in the edit screen.

I like to display the SQL query, so I will select the Above the preview option under Show SQL query and then select the Show the SQL query checkbox that follows it.

If you don't check the Show the SQL query option, it doesn't matter what you select for above or below the preview, and if you expect to see the SQL queries and don't, it is likely that you set one option and not the other. Showing the SQL query can be confusing at first, but after a while, you'll find it handy to figure out what is going on, especially if you have relationships (or should have relationships and don't realize it). And, of course, if you can't read the query, you can always e-mail me for a translation to English.

The next option, Show performance statistics, is handy when trying to figure out why a Views-generated page is loading slowly. But usually, this isn't an issue you'd be thinking of, so I'd leave it off. You want to focus on getting the right information to display exactly the way you want without thinking about the performance. If we later decide it's too slow, the developer we'll assign to it will use this information and turn the option on in development.

The same is true about Show other queries run during render during live preview. This information is handy to figure out performance issues and occasionally a display formatting issue during theming, but it isn't something you as a nonprogrammer should be worried about. Seeing all the extra queries can be confusing and intimidating, yet it doesn't really offer you any help creating a view.

"Oh, don't forget to click on Save configuration if you change any settings. I don't know how many times I've forgotten to save a configuration change in Drupal and then wondered why my change hasn't stuck. Does this help?"

"Thanks Jim, that is great. I owe you a coffee next time we get together."

Hanging up the phone, Lynn said, "What do you think, Jackson? Let's start off by creating a property maintenance page for our salespeople to use. I think I'll get a quick win by modifying one of Drupal's core views."
Adapting an existing View

Lynn will use her knowledge from using Views on her existing Drupal site, moving quickly, but we'll come back to really dig into the view edit screen in later chapters when she's building views from scratch and improving them in successive revisions.

The existing content page provided by Views is general purpose and offers lots of options, and not all these options are appropriate for all content editors. This page looks like this:

![Drupal's standard content listing page](image)

Lynn started creating her property maintenance page by going to the Views listing page (Manage | Structure | Views) and selecting Duplicate from the OPERATIONS pull-down menu on the right-hand side of this row. On the next screen, she named the view Property Maintenance and clicked on the Duplicate button. When the view edit screen appeared, she was ready to adapt it to her needs. First, she selected the Page display, assuming the Always show the master (default) display setting was already selected; otherwise, the Page display will be selected by default as it is the only display in this view.

Remember that any change made in the view edit page isn't saved until you click on the Save button. Also, unsaved changes won't show up when the page/block is displayed. If you make a change, look at it using another browser or tab, and if you don't see the change reflected, it is likely that you didn't save the change you just made.
Editing the Property Maintenance view

Starting with the left-hand side column of the view edit screen, Lynn changed the title by clicking on the Content link next to the Title label. She changed the title to Property Maintenance. Moving down the column, Lynn decided that the table display and settings were okay on the original screen and skipped them.
Under the **FIELDS** section, Lynn decided to delete the **Content: Node operations bulk form**, **Content: Type (Content Type)**, and **(author) User: Name (Author)** fields/columns as they weren't useful to the real estate salespeople who would be using this page. To do this, she clicked on **Content: Node operations bulk form** and then on the **Remove** link at the bottom of the **Configure** field modal that appeared. She repeated the removing of the field for the **Content: Type (Content Type)** and **(author) User: Name (Author)** fields. Lynn noted that the username field appeared to be the only field reference to the author entity, so she could delete the relationship later.

Moving on to **FILTER CRITERIA**, Lynn was a bit confused by the first two filters. When she clicked on **Content: Published status or admin user**, the description said **Filters out unpublished content if the current user cannot view it.** "This seems reasonable, let's keep this filter," she thought, and she clicked on **Cancel**. Next was **Content: Publishing status (grouped)**, an exposed filter that lets the user filter by either published or unpublished. This seemed useful, so Lynn kept it and clicked on **Cancel**. The next filter, **Content: Type (exposed)**, is necessary but shouldn't be selectable by the user, so Lynn clicked on it to edit the filter, unselected the **Expose this filter to visitors** option, and selected just the **Property** content type, making the filter only select content that are properties. The next filter, **Content: Title (exposed)**, is handy, so Lynn left it as is. The final filter, **Content: Translation language (exposed)**, isn't needed as Lynn's site isn't multilingual, so Lynn deleted the filter.

Moving on to the center column of the view edit page, under the **PAGE SETTINGS** heading, Lynn changed the path for the view to **/admin/property-maintenance** by clicking on the existing **/admin/content/node** path, making the change, and clicking on the **Apply** button.

Next in this column was the menu setting. Lynn doesn't want the property maintenance page to be part of the administration content page, so she clicked on **Tab: Content** and changed the menu type to **Normal** menu entry. This changed the fields displayed on the right-hand side of the modal, so Lynn changed the **Menu** link title to **Property Maintenance**, left the description blank, and left **Show as expanded** unselected. In the **Parent** pull-down menu, she selected the <Tools> menu.

![Tools is the default Drupal menu for site tools that is only shown to authorized users, who are logged into the site, and can view the page linked to, which real estate salespeople will be able to view.](image)

She left the weight at **-10**, planning on reorganizing this menu when she has most of it configured. As this is the last option, she clicked on **Apply** to exit the modal.
The last setting in the **PAGE SETTINGS** section is **Access**. Lynn knew she needed to change the required permission as she didn't plan on giving real estate salespeople access to the main content page, but she wasn't sure which permission to give them. Looking through the permissions page (the **People | Permissions** tab), Lynn didn't see any permissions that made sense for who should be able to see this maintenance page. So, she clicked on the **Permission** link in the center column of the view edit page and changed the **Access** value from **Permission** to **Role**, and when she clicked on the **Apply (all displays)** button, she could select the role(s) she wanted to be able to see on this page. She selected the **Administrator**, **Real Estate Salesperson**, and **Office Administrator** roles.

One way to test access while you develop is to use a second browser and log in as the other kind of user. A common mistake in Drupal is to see content while logged in as an administrator that can't be seen by other users. This can also be done using a second tab opened in "incognito" mode, but I find it easier to use a different browser (for example, Chrome and Firefox). You can even have three browsers open to the same page to test a third kind of user.

Continuing down the column, Lynn decided she didn't need a header or footer on this administration page at least for now, but she did want to change the **NO RESULTS BEHAVIOR** message. Drupal has a text message defined, so she clicked on the **Global: Unfiltered text (Global: Unfiltered text)** link, changed the **Content** field to **No properties meeting your filter criteria are available.**, and clicked on the **Apply (all displays)** button.

The final section, **PAGER**, seemed fine, so Lynn skipped over it and moved to the third column of the view edit page, **ADVANCED SETTINGS**. As Lynn had changed the setting to always show the advanced settings, Lynn noticed that there was a relationship for author. As she had deleted displaying the author name, there wasn't any reason to keep the relationship because she wasn't using any of the author's details. She clicked on the **author** link and then on the **Remove** link at the bottom of the modal. Reviewing the results of the live preview, Lynn was satisfied and clicked on the **Save** button to save her modified view.

There is a maxim in computers, **Save Early, Save Often**. As you develop or modify your view, when you reach a point where your progress so far is okay, click on the **Save** button. Then, if you make a terrible mistake in the next change, you can click on the **Cancel** button and then click on **Edit** to resume from where you last saved.
Before saving the view, the result looked similar to the following screen:

The resulting Property Maintenance view edit screen with all the changes

**Debugging – Live Preview is your friend**

Assuming you enabled Live Preview in your Views settings earlier in the chapter, as you are building your view, Views will show what will be displayed. Formatting and some JavaScript displays, such as Google mapping, can't be displayed in Live Preview, but to debug, you generally don't need them.
Many views' challenges are getting the data that you want to display or getting data to be displayed the way you want. Many views are created using the fields content display. Often, you will see fields that you don't want displayed when reviewing Live Preview because you didn't check the Exclude from display option in the field configuration. Or, you will select a field from the Add fields list that isn't actually the field you want to display the data you want—for instance, do you want article tags or article tags (field_tags: delta)? Sometimes you have to just try one and see what happens. If it isn't the right option, delete the field and try another. Experience will guide you as you use Views, but even the most experienced site builders wonder what some field or field option does in the context of the view they are building.

Remember to save the view before you experiment with this next idea. Then, if it doesn't work out, you can just click on Cancel and not lose all the previous work you put in. If you disabled Live Preview, hopefully, you have decided to go back and enable it; seeing the output and looking at the generated SQL queries is really very useful in trying to figure out what might be going wrong.

"Okay, Jackson, I see that a lot of what I knew from the previous versions of Views applies to the version in Drupal 8. Now that I've quickly gone through the edit screen to modify a core view, let's get serious and really learn the ins and outs of this version of Views."

Summary

In this chapter we covered the Views administration page, where you can add, delete, edit, and duplicate views. Then, we reviewed all the general Views module settings. Finally, we modified a core view, quickly going through several configuration options, which will be covered in more detail in the rest of the book. If you have used Views in older versions of Drupal, you should feel comfortable. If this is your first introduction to Views, don't panic that we glossed over a lot or if you felt lost. We'll come back in later chapters and go through the entire Views edit page in much more detail.

In the next chapter, we'll start the in-depth exploration of Views, beginning with building a simple property listing that most real estate sites might have. This listing will get successive refinements in later chapters to make it very powerful, but for now, it is a simple list. This will lead to defining some Views terms, such as View Types, displays, Display Formats, and Display Contents. With the terminology established, we'll end with an interesting view that displays selected events from Drupal's Watchdog log to users that might normally not be able to see them.
Where to buy this book
You can buy Mastering Drupal 8 Views from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.
Click here for ordering and shipping details.