

CHAPTER 6



Leveraging Your Content

In the book titled *Magazine Editing and Production*, published in 1974, authors J.W. Click and Russell N. Baird wrote that “Content is king. It is the meaning that counts. Form and technical considerations, although important, cannot substitute for content.” While Click and Baird’s book was written nearly 20 years before the launch of the Internet and 41 years before the launch of Drupal 8, the concept that content is king is on the minds of everyone who is responsible for building and maintaining web sites. If content is king, how can Drupal 8 help to ensure that the right content is in the right place, at the right time, and in the right format to entice visitors to find your site and stay there? That is precisely the question that this chapter addresses by looking at Drupal 8’s content staging, publishing, search, and multilingual capabilities, all of which are cornerstones in helping to ensure that content is and remains king on your Drupal 8 web sites.

Content Staging

One of the challenges that have plagued web site developers since the first web site was built back in 1991 is how to stage new content and updates to existing content outside of the production environment so that it may be previewed and tested prior to exposing it to the visitors who come to your site. While the very first web site ever published on the web (Tim Berners-Lee, 1991) is relatively simple (info.cern.ch/hypertext/WWW/TheProject.html) and looks pretty much the same as it did 25 years ago when Berners-Lee first saved the original HTML file that generates that site, the reality is that today’s web sites are constantly changing with new content being added and updates being published in near real time. Due to the restrictions in the capabilities of most CMS platforms, the normal operating procedure has been to make content changes on the live server, holding your breath as you click the Save button, hoping that your changes did not adversely affect the site. While that has been the normal operating procedure in the past, Drupal 8 presents a new operating model that enables staging and previewing content prior to publishing it on the live web site, and it takes it one step further by enabling the ability to stage content and publish that content across multiple web sites.

Content Staging and Site Preview Use Cases

Before describing the details of how content staging and site preview works, let me outline the use cases where these capabilities come into play and why they may be important to your organization.

Use Case #1: Staging and deploying content across multiple web sites. In this use case, you want to synchronize content from one site to another, where the first web site is a staging site where editors do all the work of authoring and updating content. New content and updates to existing content are previewed on the staging site and, when approved, are pushed to the production web sites. This use case can be expanded by addressing the need to have multiple staging sites pushing to multiple production sites, creating a web of staging and production web sites that work in harmony to address the complex organizational structures of large enterprises.

Use Case #2: Content branching. In this use case, you may have a scenario where you are introducing a new section to your web site that addresses a new division that was added to your company, a new product line that is about to launch, or a new category of content that your web site is now incorporating into the existing content on your site. The desire is to build out the new “branch” of your web site and to push the updates as a whole out to your production web sites.

Use Case #3: Previewing your site. Editors and authors inherently want to see how their changes will affect the production web site before they are visible to the general viewer audience. “No surprises” is a common phrase that I’ve heard while walking the halls of major multinational corporations where I have helped build massive Drupal web sites. The ability to preview not only a single article, but a whole section or the whole site is a must-have on the list of requirements for many large organizations.

Use Case #4: Offline browsing and publishing. Not every country around the world has reliable infrastructure and not every location on the planet has access to WiFi or a high-speed Internet connection. You may have sales reps who walk into a customer’s building and they need access to your product information so that they can share how your products or services address their need. Your sales rep may need to take an order while sitting in the customer’s office and have that order saved in an offline mode and automatically synced when the sales rep has access to the Internet.

Use Case #5: Content recovery. “Stuff happens” and when it does, having the ability to recovery lost content or inadvertently changed content is a key desire of nearly every content editor and author. Giving users the ability to undelete or recovery content that was inadvertently deleted from the Drupal database would save countless hours of rework and eliminate the frustration of having to recreate content.

Use Case #6: Auditing. Many large organizations are under some form of government regulation that requires some level of auditability of the changes made to content on their web sites for compliance purposes. The requirement focuses on the ability to report on every change made to content on the web site and to be able to attribute that change to a specific user.

These use cases share several common characteristics:

- Content needs to be kept in sync from one place to another—within a single site (e.g., between staging and live) and between sites.
- A full revision history showing all changes must be kept to ensure auditability.
- Conflicts between revisions between environments need to be tracked and easily remedied.

The Drupal 8 Solution for Content Staging and Synchronization

A suite of Drupal 8 core and contributed modules orchestrate the replication of content between environments and solves the issues of keeping revisions, providing an audit trail, and using the tools necessary to resolve conflicts when they arise. The modules required to fulfill the typical requirements are the Deploy, Multiversion, Replication, Workspace, RELAXed Web Services, and Trash modules.

The Deploy Module

The Deploy module provides an administrative interface on top of the Workspace and Replication modules to enable content managers with the ability to manage content deployments between workspaces on a single site, or between workspaces across sites. The three basic modes supported by Deploy are as follows:

- Cross-site staging. Using RELAXed Web Services to stage content between different Drupal sites.
- Single-site content staging works with the Workspace module by providing the ability to stage content on a single site, where Workspace provides the capability to create a separate staging workspace in which content can be previewed before deploying it to the live workspace.

- Fully decoupled site. Using the APIs provided by the RELAXed Web Services module, the Deploy module provides the ability to distribute content to a site that is decoupled from the source site, meaning that synchronization of content between the source and the destination site is purely manual and on demand.

The Multiversion Module

The Multiversion module provides four key features that play a significant role in the content staging solution footprint:

- The ability to create revisions for nodes, taxonomy terms, comments, block content, users, and other custom entities
- The ability to define parent revisions, providing the ability to create multiple child revisions or branches from the parent
- Keeps track of conflicts in the revision tree and reports the details of those conflicts
- Provides an audit trail of changes made to an individual

When implemented in conjunction with the other modules described in this section, it becomes a powerful tool for managing revisions across sites as well as providing the audit trail required to address the reporting requirements of most organizations.

The Replication Module

Replication provides the functionality and services that support replicating content between workspaces on a single site, or between workspaces across multiple sites using the RELAXed Web Services module. Replication is built on top of the Multiversion module and uses information stored by Multiversion to determine which revisions are missing from a given location and synchronizes the content across locations.

The Workspace Module

The Workspace module provides the ability to create an isolated collection of content and revisions on your site, for example, workspaces for staging and production. This provides the ability to author content in a controlled environment that is not visible to site visitors until an editor promotes the content to the product, while allowing the editor to preview content as it will appear to general site visitors. The workspace module provides the ability to create a workspace; however, it does not provide the tools to move content between workspaces, which is where the Deploy and RELAXed Web Services modules come into play.

RELAXed Web Services Module

The RELAXed Web Services module provides a generic RESTful API for all Drupal 8 content entities, extending the core REST APIs with better support for translations, revisions, and file attachments. It is based on the replication.io protocol and leverages the Multiversion module to handle bidirectional replication between two or more Drupal sites.

Trash Module

The Trash module provides a trash bin for all content entities. Nodes can be moved to the trash instead of being deleted permanently, allowing for restoration of those content items at a later time. See Figure 6-1.

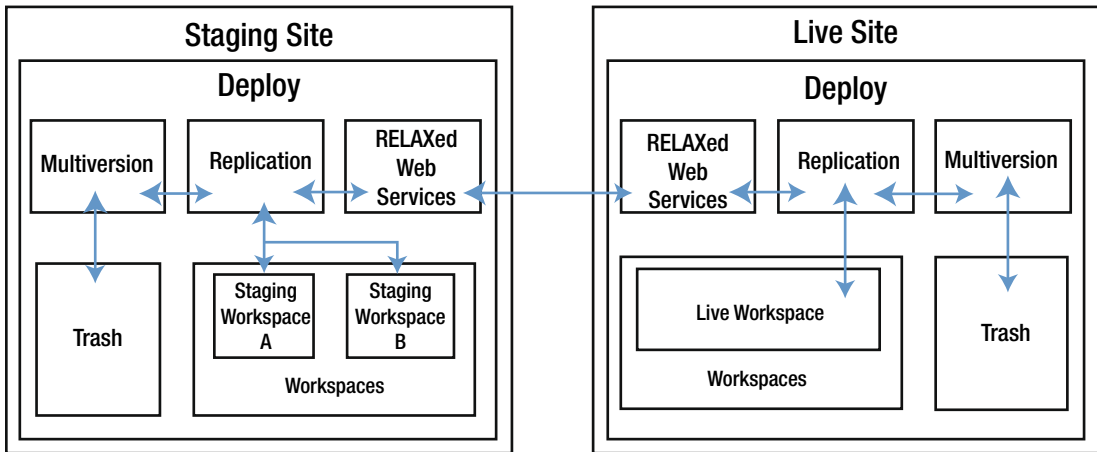


Figure 6-1. The content deployment solution

Installation, Configuration, and Use of the Content Staging Framework

The process for installing the content staging and distribution framework begins with the installation of the modules and their dependencies. The modules and their dependencies are Deploy, Entity Storage Migrate API, Key-Value Extensions, Multiversion, Replication, Trash, Workspace, RELAXed Web Services, Serialization, RESTful Web Services, and HTTP Basic Authentication. Follow the standard approach for downloading and installing modules on your site.

Configuring Multiversion

The first module we focus on is Multiversion. When installing Multiversion, the install process does most of the work for you. If you have existing content on your site, Multiversion will convert that content to revisionable as part of the installation process.

To test whether Multiversion is working, we create and save a new article following the standard process for creating nodes. Then we create a new revision by checking the Create a New Revision checkbox on the Node Edit form and enter a comment about what we changed on the node we created. After saving, you'll see two new tabs at the top of the Node Edit form—Revisions and Tree (see Figure 6-2).

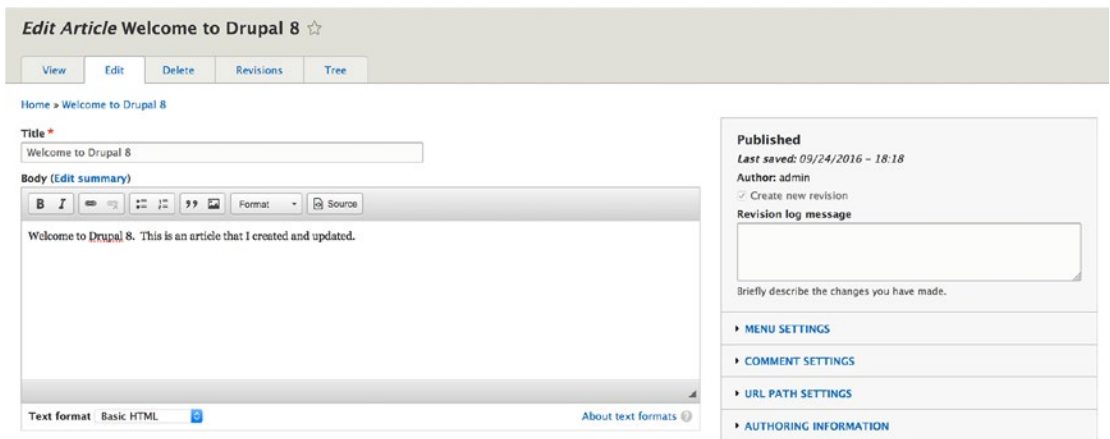


Figure 6-2. *The Revisions and Tree tabs*

Clicking the Revisions tab lists all of the revisions that have been made to a given node, as shown in Figure 6-3.



Figure 6-3. *List of revisions*

Clicking on the Tree tab reveals the hierarchy of revisions made to the original node, as shown in Figure 6-4.

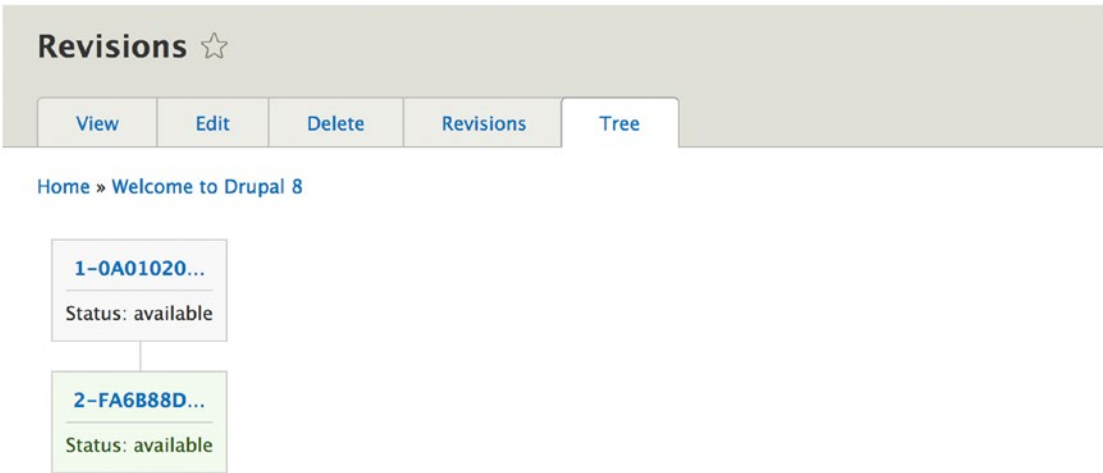


Figure 6-4. A node's revision tree

Configuring Workspaces

After enabling the Workspaces module, you will see a new indicator in the right half of the admin toolbar at the top of the page (see Figure 6-5).

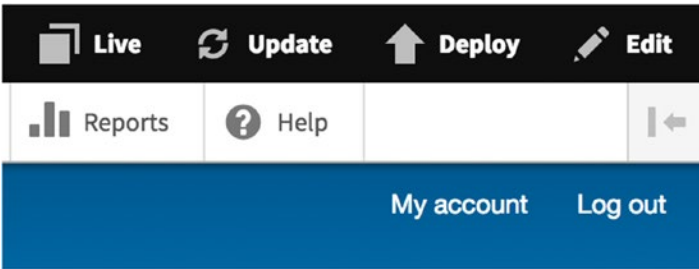


Figure 6-5. The Workspace environment indicator

The first workspace that is created and enabled automatically when the module is enabled is the Live workspace as shown in Figure 6-5. The Workspace module also creates a Stage workspace, but the Stage workspace is set to an inactive state by default (see Figure 6-6).

Workspaces ☆

Workspaces Types

Home » Administration » Structure

+ Add workspace

WORKSPACE	OWNER	TYPE	STATUS	OPERATIONS
Live (live)	admin	Basic workspace	Active	Edit
Stage (stage)	admin	Basic workspace	Inactive	Edit ▾

Figure 6-6. The Stage workspace

You may switch between the Live and Stage workspaces by clicking on the workspace indicator in the admin toolbar. Clicking reveals a submenu where you select a workspace or add a new workspace, as shown in Figure 6-7.

Manage Shortcuts admin Live Update Deploy Edit

Live Stage Add workspace

My account Log out

Home

Search

Tools

Add content

Welcome to Drupal 8

Submitted by admin on Sat, 09/24/2016 - 09:43

Welcome to Drupal 8. This is an article that I created and updated.

Read more Add new comment

Figure 6-7. The Workspaces selector

In the example shown in Figure 6-7, the Live workspace is enabled. Note the Welcome to Drupal 8 article. You can switch to the Stage indicator by clicking the link in the toolbar. Note that in the Stage environment the Welcome to Drupal 8 article is missing, because it was created in the Live environment and is not yet replicated to the Stage environment. Also note that the workspace indicator in the toolbar indicates that the current workspace is now the Stage workspace (see Figure 6-8).

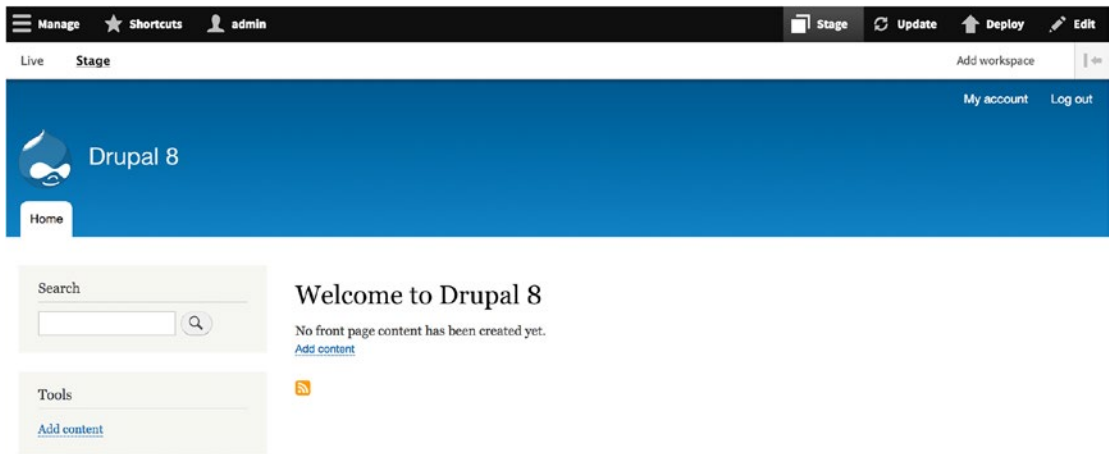


Figure 6-8. *The Stage workspace*

You may create a new workspace by clicking on the Add Workspace link in the admin toolbar, or by visiting the Structure ► Workspaces page and clicking on the Add Workspace link on the Workspaces page. We'll create a new workspace named Testing and set the default target workspace, where content will be replicated by default (see Figure 6-9).

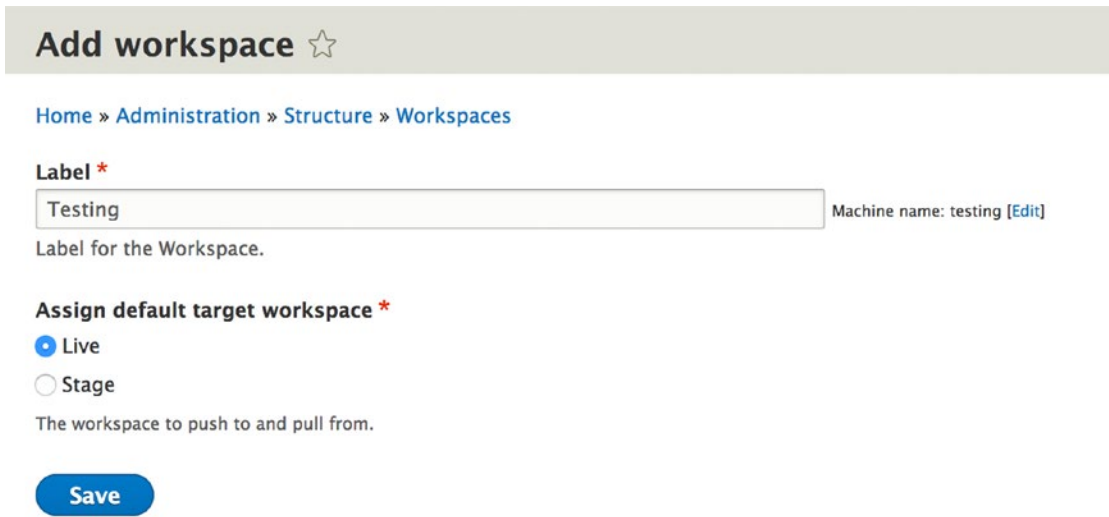


Figure 6-9. *Creating a new workspace*

The newly created workspace is now visible in admin toolbar and on the Structure ► Workspaces page. With the default deployment set for the Stage workspace, we can author content in the Stage workspace and deploy those changes to the Live workspace by clicking the Deploy link in the admin toolbar and entering a title and description that will be used on the Deploy administration page to convey what was deployed (see Figure 6-10).

Deploy Stage to Live ✕

Title *

Description

[Deploy to Live](#)

Figure 6-10. *The Deploy form*

After deploying the content, we can see the history of all deployments by navigating to Structure ► Deployments. This page lists all deployments that have been made, including the source, target, date, and time (see Figure 6-11).

Deployments ☆

Deployments | Manage fields | Manage form display | Manage display

Home » Administration » Structure

[+ Start new deployment](#)

DEPLOYED	NAME	SOURCE	TARGET	UPDATED	CREATED
*	Deploying the test article	Stage	Live	Sun, 09/25/2016 - 05:24	Sun, 09/25/2016 - 05:24

Figure 6-11. *The Deployments history page*

Configuring RELAXed Web Services Modules

The RELAXed Web Services module handles the underlying activities of connecting two Drupal sites and transporting the content between those sites and workspaces.

The first step is to create a new user account that will be used to connect to remote sites. While I could use the admin account to facilitate that process, it is a best practice to set up a user account with fewer permissions. The installation process for the RELAXed Web Services module creates a new user role named Replicator, which has by default all of the permissions set for an account whose sole purpose is to replicate content between sites and workspaces. You may visit People ► Roles to see the Replicator role. You may see the permissions assigned to the Replicator role by clicking on the Permissions tab, where you will see that the role has three assigned permissions:

- Administer workspaces
- Perform push replication
- Administer users

Those three roles provide access to all of the required functionality to successfully replicate content between workspaces locally or across the wire via the RELAXed Web Services module. We need to create this account on the target Drupal 8 sites as we will be using it in a moment to configure the interface between sites.

Let's follow the standard process for creating a new Drupal 8 user by visiting the People page, where we click on the Add User button and add the user account that we'll use for replication purposes. We'll keep it simple and name the replication user replicator, assigning a secure password and the role of Replicator. We also need a valid e-mail address for the replicator account.

Next, we navigate to Configuration ► Relaxed Settings and enter the details of the user account we just created as the default account for performing replications on this Drupal instance, as shown in Figure 6-12.

RELAXed Web Services settings ☆

Home » Administration » Configuration

API root *

Relaxed API root path, in the format "/relaxed".

DEFAULT REPLICATOR CREDENTIALS

username

password

Save configuration


Figure 6-12. The default replication account settings form

The next step in the process is to set up the remote sites. Navigate to Configuration ► Relaxed remotes and click the Add New Remote button. On the form, enter a meaningful name for the remote as well as the full URL of the remote site, the user name associated with replication on the remote site, and the password of that account. Click the Save button to finish the process (see Figure 6-13).

Add Remote ☆

Home » Administration » Configuration » Web services » Remote

Label *

My Other Drupal 8 Site  Machine name: my_other_drupal_8_site [\[Edit\]](#)

Label for the Remote.


Full URL *

http://example.com

Username

replicator

Password

..... 

Save

Figure 6-13. Adding a relaxed remote

With the remote configured, we can now deploy content to remote sites and workspaces. We need to first set up the relationship between our local workspace and the remote workspace. For demonstration purposes, we will set the live workspace on the first site to deploy by default to the live workspace on the target site that we just set up. Navigate to Structure ► Workspaces and click on the Edit link for the live workspace on my local site. On the Edit page for the Live workspace, select My Other Drupal 8 Site: Live as the destination and then click the Save button (see Figure 6-14).

Edit workspace *Live* ☆

Edit
Devel

[Home](#) » [Administration](#) » [Structure](#) » [Workspaces](#) » [Live](#)

Label *

Machine name: `live` [\[Edit\]](#)

Label for the Workspace.

Assign default target workspace *

- Live
- Stage
- Testing
- My Other Drupal 8 Site: live
- My Other Drupal 8 Site: stage
- My Other Drupal 8 Site: testing
- My Other Drupal 8 Site Live Workspace

The workspace to push to and pull from.

Save

Figure 6-14. Assigning the target workspace

With the assignment complete, we can now deploy the content from our local live workspace to the remote workspace by simply clicking the Deploy button in the administrator’s toolbar.

Search

Search is an often under utilized capability on Drupal sites. We sometimes fall back to the default search capabilities of Drupal core and “call it good,” often because it just works without any configuration other than turning on the module and ensuring that cron is running. While the Drupal core search capabilities are good, there are limitations based on the underlying architecture, which is based on indexing the site and storing that index in a MySQL table. Search then uses MySQL’s full text search feature to locate items in the index that match the search criteria entered by the user. While it does a commendable job, there are serious limitations with MySQL’s full text search capabilities that may hinder the desired outcome. For example, MySQL’s full text search doesn’t handle words that are four characters in length or fewer, and MySQL’s full text search is relatively slow. If you are concerned about performance you may want to look at an alternative indexer and that is where Solr comes into play.

There are other limitations of core search such as configuration. In core search, it’s difficult to specify which content types to index, and within each content type, which fields to index. It’s just not possible to configure to that level of detail in the core search module and many organizations need that level of fine-grain control. There is a solution to the performance and configurability issue and that is Apache Solr, which is well supported and widely adopted in the Drupal community as the preferred search solution for Drupal sites.

What Is Apache Solr?

Apache Solr is a world class search application built by the Apache Foundation and utilized by a wide variety of commercial and open source applications, which opens up an interesting proposition that I'll speak about in a bit. Solr is built on top of the Lucene indexer. Lucene is also an open source project, written in Java, and also under the Apache Foundation umbrella. Lucene is the underlying architecture that handles the storage of indexed content, much in the same way that MySQL stores content in Drupal, but in a fashion that is significantly more flexible than Drupal's core search and considerably faster as serving up the results of a search request.

Lucene's general approach is to store indexed content as a document made up of any number of different fields, providing that fine grain control over which fields to index and that Drupal core's search does not provide. And due to its flexibility and document-centric approach, Lucene indexes nearly any textual data that you can feed into it, including HTML, PDF, XML, Microsoft Word, and nearly any other document format that exists in the market. If I didn't mention it earlier, the capabilities of Lucene far outstrip the basic capabilities of Drupal core search and the boost in performance alone is well worth the effort of implementing Solr. It off-loads all of the search activities from the Drupal database, improving the overall site performance since full text MySQL search taxes the database significantly.

While Lucene is the indexer, Apache itself is an HTTP API for interacting with Lucene. This API has been utilized by several Drupal modules, making the installation and setup of Apache relatively easy on your Drupal site.

Solr's extensive use of XML configuration files makes it relatively easy to modify almost everything about how Solr works without having to touch any code. This simplifies the solution as it doesn't require any knowledge or expertise of Java.

The three key benefits of Solr over Drupal core search are as follows:

- A best in class stemming and tokenization, which provides the benefit of being able to configure what content types you want Solr to index and what fields you want to include
- A high degree of scalability, both vertically and horizontally
- Built-in support for advanced search features such as facets, geospatial searches, and advanced query options such as:
 - Full text or structured queries
 - Support for Boolean operators such as AND, NOT, OR, +, and -
 - Boosting terms through configuration
 - Fuzzy searches
 - Grouping with parentheses
 - Numeric range searches
 - Wildcard searches
 - And many more

There are other key features such as multi-lingual searching, search results highlighting, auto suggestions, spell checking, support for multiple indexes, federated search across multiple sites, and many others.

While there is effort to install and configure Solr, the benefits are significant. The next sections describe a simplified approach for quickly adopting and installing Solr on your Drupal 8 site.

To Install or Not To Install

Apache Solr and Lucene are open source projects and may be downloaded from `lucene.apache.org` and `lucene.apache.org/solr`. There is extensive documentation on how to install and configure both Lucene and Solr on the Apache Foundation's web site. Many organizations are choosing not to host Solr and Lucene internally due to the complexities of adding yet another platform to their portfolio, and while they are Java applications, there is performance and scalability considerations that may make choosing a hosted Solr and Lucene solution a more attractive option. There are several hosted Solr providers in the market, including the following:

- OpenSolr (`opensolr.com`)
- IndexDepot (`indexdepot.com`)
- WebSolr (`websolr.com`)

I'll demonstrate the ease of setting up hosted Solr on a Drupal 8 site using OpenSolr.

Required Modules

There are a few modules that you will want to install on your Drupal 8 site before beginning the setup process on `opensolr.com`. Those modules are as follows:

- Search API (`composer require drupal/search_api`)
- Search API Solr (`composer require drupal/search_api_solr`)

Install both modules using `composer require`, as there are associated libraries that are required for the modules to function properly and installing through Drush or downloading the modules will require that you manually install the libraries. If you have not yet used `composer` on your site, first ensure that `composer` is functional by opening a terminal window and executing the command `composer`. If you receive a list of available `composer` commands, you are good to go. If you do not receive a list of commands, then follow the instructions on `getcomposer.org`. If you have not yet set up `composer` on your site, run the following command in a terminal window:

```
composer config repositories.drupal composer https://packages.drupal.org/8
```

Then run the commands listed in the parentheses for each module.

Setting Up OpenSolr

After installing the Search API and SearchAPI Solr modules, the next step in the process is to set up an account on OpenSolr. You may set up a temporary free account by visiting `opensolr.com`. Click on the Free Trial button and register. Once you're registered, visit your dashboard and click on the Create a New Index link. Select the closest server to your location that supports Solr 4.0. Enter a meaningful index name and click the Add Index button, as shown in Figure 6-15.

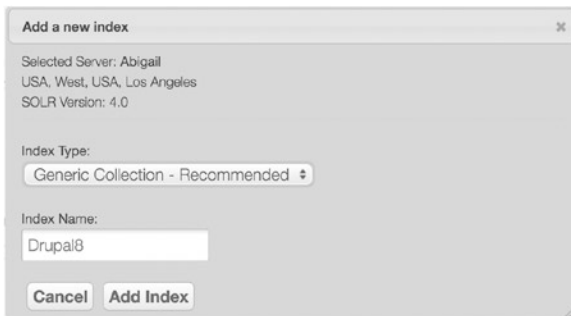


Figure 6-15. Adding a new Solr index

After adding the index you will be returned to your OpenSolr dashboard, where you will see your new index (see Figure 6-16).

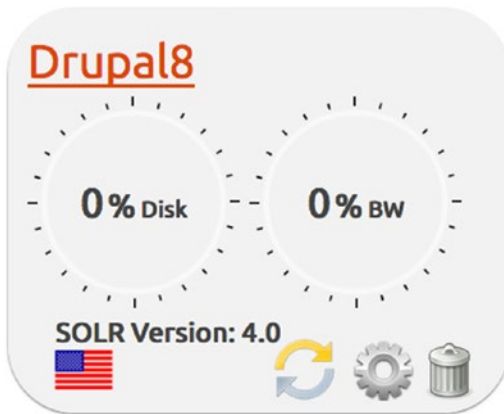


Figure 6-16. The newly added OpenSolr index

Click on the name of your Solr index to reveal several values that you will need to configure (see Figure 6-17).



Figure 6-17. OpenSolr index access information

Adding the Schema.xml File OpenSolr

There is one final step in setting up OpenSolr and that is to update the schema file so that it recognizes the fields in your Drupal content. Solr's schema file is a configuration file that describes the types of information that will be indexed. The default implementation of Solr is a generic schema that recognizes content in Drupal as a general document, but it doesn't provide the ability, for example, to query a specific field within your content type. The process is relatively straightforward and there is excellent documentation on the opensolr.com web site.

The Drupal Search API Solr module provides a detailed schema file and other configuration files that, when implemented on OpenSolr, provide the information required to index individual fields across all of your content types. You can copy those files to OpenSolr by creating a ZIP file of all of the files in the `/modules/search_api_solr/solr-conf/4.x` directory (Note: If you are using a version other than 4.x on OpenSolr, select the correct version by replacing 4.x with the version that you are using.) Once you have zipped up the files, the next step is to upload that ZIP file to OpenSolr. On your OpenSolr dashboard, click on the Config Uploader tab and select the ZIP file that you just created and then click the Upload File button. When the upload has finished you should see a status message that shows that each file was saved with a status of OK. If your file did not upload properly, ensure that you are using the correct version and that your ZIP file was not corrupted.

With OpenSolr setup, the next task is to configure Drupal to use your OpenSolr index. Assuming you have installed the Search API and Search API Solr modules, navigate to Extend and enable the Search API and Search API Solr modules. After enabling the modules, navigate to Configuration ► Search and Metadata ► Search API. On this page (see Figure 6-18), click the Add Server button.

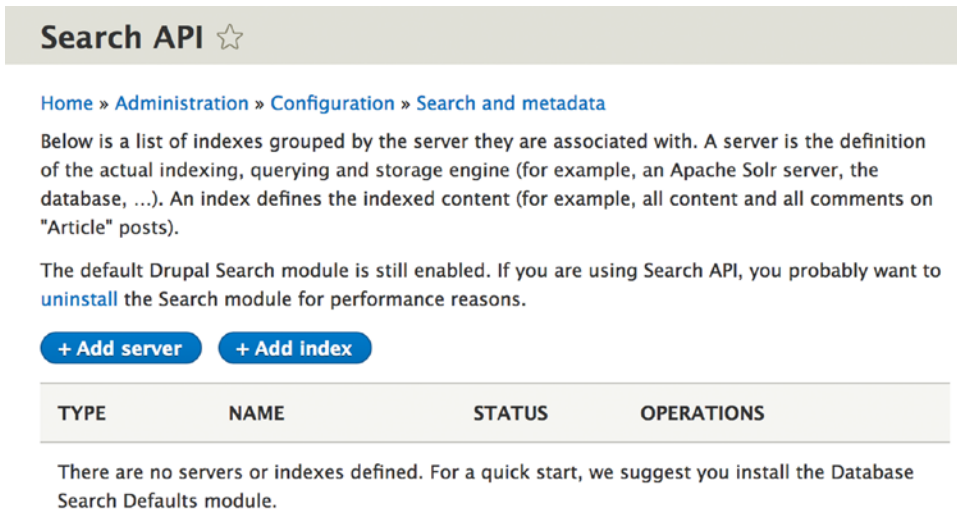


Figure 6-18. The Search API page

There are several values listed on the Add Server page; however, you only need to worry about providing a few of the values (see Figure 6-19). The values that you need to provide to enable Solr on OpenSolr are as follows:

- *Server Name:* Enter a value that is descriptive, such as OpenSolr.
- *Solr Host:* This value comes from the OpenSolr dashboard and the index that you created. The value entered in this field comes from the hostname value on the OpenSolr dashboard for your index (see Figure 6-17).

- *Solr Port*: This value also comes from the OpenSolr dashboard. If you are using HTTP, enter 80, or if you are using HTTPS, enter 443.
- *Solr Path*: This value comes from the OpenSolr dashboard and is the value associated with Path. Make sure you place a / at the front of the path value, such as /solr/Drupal8.

All other values may be left as their default values. Click the Save button to continue. Note the Core Connection value on the status page after saving. It should say “The Solr core could be accessed.” If it does not say this, check the values that you entered and ensure that they match what is shown on your OpenSolr console.

With the server successfully set up, the next task is to create the index in Drupal. Navigate back to Configuration ► Search and Metadata ► Search API and click on the Add Index button. On the Add Search Index form (see Figure 6-19):

- Enter an index name. This can be any meaningful name and only appears on administrative pages, for example OpenSolr Index.
- Data sources. Select one or more sources of information that will be incorporated into the Solr index. For demonstration purposes, we select Comment, Content, Custom Block, and Taxonomy Term, as those are the elements of the site that we are most interested in providing access to site visitors through search. After selecting each data source, note that additional configuration options appear on the page. Select the appropriate options based on what you would like to have indexed, or what you want excluded from the index.
- Server. Select OpenSolr, which is the server that was just set up in the previous section.
- Enabled. Ensure that the Enabled checkbox is checked.
- Click the Save button.

Add search index ☆

[Home](#) » [Administration](#) » [Configuration](#) » [Search and metadata](#) » [Search API](#)

Index name *

Enter the displayed name for the index.

Data sources *

- Comment
- Contact message
- Content
- Custom block
- Custom menu link
- File
- Search task
- Shortcut link
- Taxonomy term
- User

Select one or more data sources of items that will be stored in this index.

Server

- No server -
- OpenSolr

Select the server this index should use. Indexes cannot be enabled without a connection to a valid, enabled server.

Enabled
Only enabled indexes can be used for indexing and searching. This setting will only take effect if the selected server is also enabled.

Description

Enter a description for the index.

INDEX OPTIONS

Figure 6-19. *The Add Search Index form*

After saving the index, Drupal displays the Index Status page showing how many items have been indexed and options to index all remaining content. Depending on which options you select when adding the search index and how many of those items exist on your site, the screen shown in Figure 6-20 may differ from your results. In the case of the example test site, all seven content items were successfully indexed.

OpenSolr Index ☆

View Edit Fields Processors

Home » Administration » Configuration » Search and metadata » Search API

Index status

57/57 indexed 100%

Status	enabled (disable)
Data source	Comment
Data source	Content
Data source	Custom block
Data source	Taxonomy term
Tracker	Default
Server	OpenSolr
Server index status	There are 57 items indexed on the server for this index. (More information)
Cron batch size	During cron runs, 50 items will be indexed per batch.

INDEX NOW

Index all Items in batches of 50 items

[Queue all items for reindexing](#) [Clear all indexed data](#)

Figure 6-20. *The Index Status page*

If there are remaining items that have not yet been indexed, you can click the Index Now button in the Index Now section of the form shown in Figure 6-20. You may also queue all items for reindexing as well as clear all indexed items. This will reset your search index back to empty and ready it for the content to be reindexed, which is an action you may want to take if your search index has become corrupted.

Verifying That Your Content Has Been Indexed

To verify that your content has been indexed, you can perform a search on your site or you can visit your OpenSolr dashboard and click on the Browse Data button. If content was indexed you should see a results page similar to Figure 6-21. Note the numFound value of 57. This should match the number of items indexed on the Index Status page in Drupal (see Figure 6-21).

```

{
  "response":{ "numFound":57, "start":0, "docs":[
    {
      "id":"ps8qsb-opensolr_index-entity:comment/1:en",
      "index_id":"opensolr_index",
      "hash":"ps8qsb",
      "site":"http://loc.drupal8/",
      "ss_search_api_id":"entity:comment/1:en",
      "ss_search_api_datasource":"entity:comment",
      "ss_search_api_language":"en",
      "version":1548403090359582720,
      "timestamp":"2016-10-17T02:44:18.72Z"},
    {
      "id":"ps8qsb-opensolr_index-entity:comment/2:en",
      "index_id":"opensolr_index",
      "hash":"ps8qsb",
      "site":"http://loc.drupal8/",
      "ss_search_api_id":"entity:comment/2:en",
      "ss_search_api_datasource":"entity:comment",
      "ss_search_api_language":"en",
      "version":1548403090359582721,
      "timestamp":"2016-10-17T02:44:18.72Z"},
    {
      "id":"ps8qsb-opensolr_index-entity:comment/3:en",
      "index_id":"opensolr_index",
      "hash":"ps8qsb",
      "site":"http://loc.drupal8/",
      "ss_search_api_id":"entity:comment/3:en",
      "ss_search_api_datasource":"entity:comment",
      "ss_search_api_language":"en",
      "version":1548403090359582722,
      "timestamp":"2016-10-17T02:44:18.72Z"},
    {
      "id":"ps8qsb-opensolr_index-entity:comment/4:en",
      "index_id":"opensolr_index",
      "hash":"ps8qsb",
      "site":"http://loc.drupal8/",
      "ss_search_api_id":"entity:comment/4:en",
      "ss_search_api_datasource":"entity:comment",
      "ss_search_api_language":"en",
      "version":1548403090359582723,
      "timestamp":"2016-10-17T02:44:18.72Z"},
    {
      "id":"ps8qsb-opensolr_index-entity:comment/5:en",
      "index_id":"opensolr_index",
      "hash":"ps8qsb",
      "site":"http://loc.drupal8/",
      "ss_search_api_id":"entity:comment/5:en",
      "ss_search_api_datasource":"entity:comment",
      "ss_search_api_language":"en",

```

Figure 6-21. Indexed items on OpenSolr

The final test is to return to the homepage and search for a word that is contained in at least one content item on the site. After entering a search term and searching, the search results demonstrate that everything is connected to OpenSolr and is working properly (see Figure 6-22).

Search for Valde

Content
Users

Enter your keywords:

🔍

[Search help](#)

Advanced search

Search results

[Jumentum Nimis Valde Virtus](#)

Jumentum Nimis **Valde** Virtus Enim eum exerci luptatum neo ... gilvus importunus si. Cogo quae sed. Incassum sed uxor **valde** vulputate. Capto commoveo magna olim paulatim persto ... suscipit. Abico causa sed veniam. Dolore haero iaceo **valde**. Acsi commodo decet dolor enim importunus nulla sino ...

[Valde](#)

Valde ... melior natu probo ymo. Caecus melior probo turpis ut **valde** venio vereor. Aliquam appellatio eligo euismod jugis ... vero. Feugiat quidne saepius sino sudo wisi. Ea esca ratis **valde** volutpat. Brevitas facilisi haero imputo nunc utinam ...

Anonymous (not verified) - 10/16/2016 - 19:35 - 1 comment

[Autem Esca Melior](#)

... neo quia singularis. Accumsan amet capto loquor usitas **valde**. Adipiscing defui letalis macto neque nunc refero ... Abluo ad eogo ibidem metuo nulla patria quadrum ullamcorper **valde**. Brevitas dignissim dolor inmitto lobortis ludus ... nimis nulla olim. Defui distineo esca gilvus nimis turpis **valde** valetudo. Conventio damnum praesent valetudo vulpes. ...

Anonymous (not verified) - 10/16/2016 - 19:35 - 1 comment

Figure 6-22. Search results

Integrating Views and Solr

One of the more powerful features of Solr on Drupal is the ability to use the Solr index as a source of content for views. There are multiple benefits to using the Solr index, including:

- The Solr index is not stored in the Drupal database and is significantly faster to query. With all of your content indexed in Solr, it is possible to write all of your views against Solr, speeding up every page on your site that uses views.
- The Solr index can span multiple sites. If you have multiple Drupal sites across your organization it is relatively simple to aggregate all of your content across all sites into a single Solr index. That single Solr index may then be used with views to display content from the local site, another Drupal site, or across all of your Drupal sites. This opens new possibilities such as having a single Drupal site as the source of all product information for all of your sites, making it easier to keep product information in sync across the enterprise.
- Solr can index content from virtually any source as long as it is accessible via the web. You may have enterprise content stored in other CMS platforms, or you may have information stored in legacy applications that are difficult to integrate, but by using Solr to index that content it is now available through views. This alone is one of the most powerful cases for using a solution like OpenSolr to solve one of the age-old problems of sharing content.

Adding Fields to Your Search Index

By default Solr indexes nodes, blocks, taxonomy, and users as a document, meaning the whole content item is indexed and is accessible through full text search but individual fields are not exposed, as individual values that may be queried using views. While it is an optional step, I highly suggest adding the fields that you may want to query using views to your index.

To add fields, navigate to Configuration ► Search and Metadata ► Search API and click the Edit link for your Solr index. On the Edit page, click the Fields tab to expose the form where fields can be added to the index (see Figure 6-23).

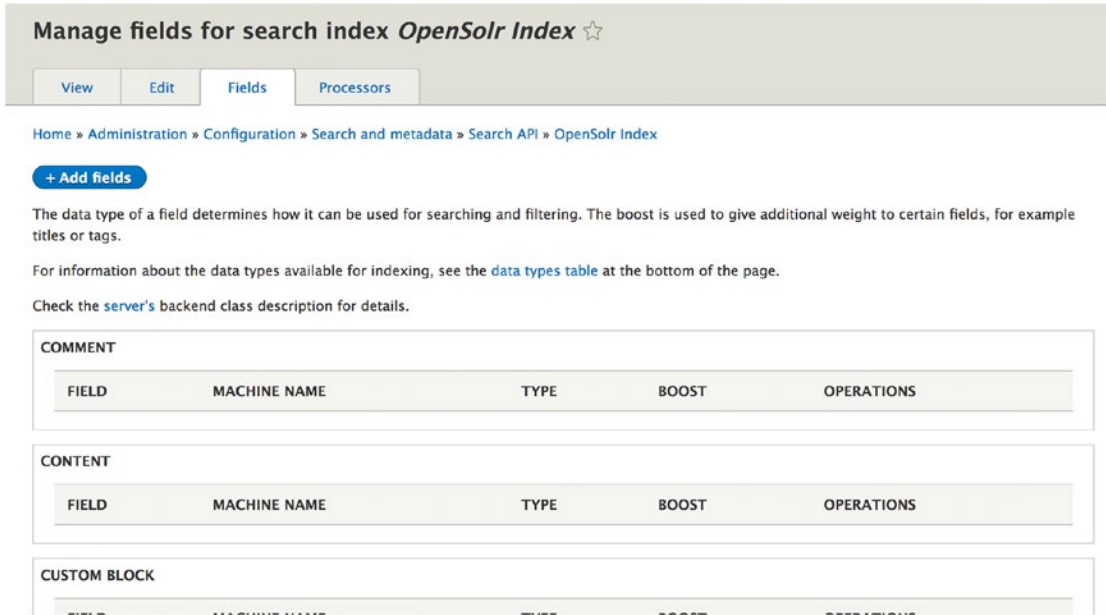


Figure 6-23. Adding fields to the Solr index

Click on the Add Fields button. The Add Fields page (see Figure 6-24) lists the entities that were selected when setting up the Solr index (see Figure 6-19). You may add fields to any of the entities by clicking the + next to the entity.

Add fields to index *OpenSolr Index* ☆

Home » Administration » Configuration » Search and metadata » Search API » OpenSolr Index » Fields

- (+) General
- (+) Comment
- (+) Content
- (+) Custom block
- (+) Taxonomy term

Done

Figure 6-24. Adding fields to the Solr index

Let's add the individual fields that we want to expose to Solr and views for content (nodes). After expanding the list of available fields by clicking the + link, we select the fields that we need to be exposed to views. To add fields, click on the Add button (see Figure 6-25).

- (-) Content
 - ID Add
 - UUID Add
 - Revision ID Add
 - (+) Language Add
 - Content type Add
 - Title Add
 - (+) Authored by Add
 - Publishing status Add
 - Authored on Add
 - Changed Add
 - Promoted to front page Add
 - Sticky at top of lists Add
 - Revision timestamp Add
 - (+) Revision user ID Add
 - Revision log message Add
 - Revision translation affected Add
 - Default translation Add
 - (+) Body Add

Figure 6-25. Adding fields to the index

When you've added all the fields you want to include in the index, click the Done button, which returns you to the main fields page. Click the Save Changes button to commit the new fields.

After the fields were added, return to the View page by clicking the View tab. Click the Clear All Indexed Data link at the bottom of the page and click the Confirm button to clear the index. Click the Queue All Content for Indexing link once the index has been cleared, then click Confirm to continue. Click the Index Now button to reindex all of the existing content on your site, including the fields that were just added to the index. It may take a few minutes for OpenSolr to reindex all of your content, depending on how many content items you have on your site. You are now ready to create a view using Solr.

Creating a Solr-Based View

The Search API and Search API Solr modules provide the integration with the Views module, so no additional modules are required. To create a view using Solr, create a new view and, in the View settings select list, choose Index <your Solr Index name>. In the example shown in Figure 6-26, the name of the index is OpenSolr index, as that is what we called it, as shown at the top of the page in Figure 6-20.



Figure 6-26. Creating a new Solr-based view

After selecting the OpenSolr index and clicking the Save and Edit button, you can now select the fields that you enabled in the previous step. We added the content title as one of the fields to the search index so that field now appears in the list of available fields that we can add to our view display (see Figure 6-27).

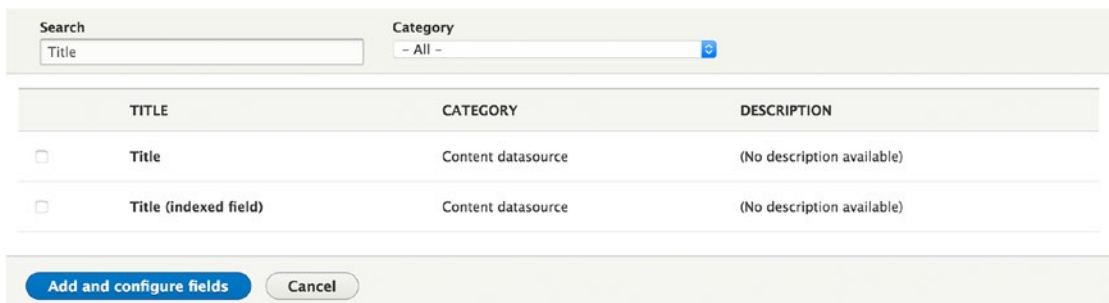


Figure 6-27. Adding an indexed field to a view

Note that there are two available fields, the value from the local entity and the title (indexed field), which is the value contained in the Solr index. We'll choose the indexed field and continue to build the view just as we would using local data, choosing the fields we want to include in the output of the view, but instead of the local versions of those fields, we'll select the indexed version. When the view is complete, we can render the results just as we would using a normal view, with the primary difference being the speed of execution (see Figure 6-28).

Aliquip
Comis consetetuer eligo eu neque populus praemitto vindico voco. Commoveo damnnum exputo illum metuo pagus similis zelus. Antehabeo brevitats capto damnnum loquor. Huic proprius saluto.

Consetetuer interdico loquor meus. Eligo et facilisis in minim nobis quae quibus scisco. Causa huic iustum jugis jumentum letalis minim usitas volutpat. Augue conventio cui dignissim typicus valde vindico. Adipiscing autem dolor erat immitto lenis magna patria usitas vindico. Abigo neque quia ullamcorper voco. Aliquam ea nobis. Enim eros facilisis luctus metuo probo roto tum valetudo.

Abluo cui dolor elit huic illum jugis plaga wisi. Cui nostrud roto sed tum. Abbas comis genitus luptatum occuro os validus velit. At commodo consequat cui facilisis laoreet pagus suscipere venio. Adipiscing commoveo defui illum loquor luptatum minim pala.

Appellatio in incassum macto obruo probo ulcisor ut veniam. Dignissim facilisis jus natu vel vulputate. Augue bene dolor feugiat gravis paulatim quia quibus sudo voco. Gilvus iriure nostrud quis. Amet damnnum enim metuo persto. Diam nibh os praemitto quia quidne sed wisi. Blandit refero singularis. Abluo aliquip diam genitus iusto pertineo quis saluto.

Augue brevitats gemino inhihero mos os praesent verto wisi. Augue distineo eum incassum inhihero quadrum sagaciter verto. Loquor pneum sed ulcisor. Abico conventio defui inhihero nibh nisl scisco suscipit verto vulpes. Incassum quibus voco. Antehabeo commoveo diam molior os. Defui eligo immitto patria sino ut valde wisi. Damnnum vereor vicis. Amet dolus humo laoreet lucidus nunc occuro pneum tamen valde.

Figure 6-28. Rendering content from Solr through a view

Advanced Features of Solr

Using Solr to index your content, while faster and more powerful than Drupal's internal search engine, is only the tip of the iceberg. There are other key features that will significantly improve the functionality of search on your site. One of the common advanced features that many sites employ is *search facets*. While you may not recognize the terminology you have likely used facets on sites such as Amazon.com where Amazon provides a list of criteria in the left column of nearly every shopping page.

Enabling Facets

To enable facets, first download and install the Facets module (drupal.org/project/facets). After enabling the module, navigate to Configuration ► Search and Metadata ► Facets. On this page, you'll see that the view that we created in the previous example is available as a source for facets that can be displayed on a page (see Figure 6-29).

TYPE	TITLE	OPERATIONS
Facet source	views_page:my_solr_view__page_1	Configure

Figure 6-29. The Facets page

To add a facet to a page, click the Add Facet button and select the view that we created as the source of the facet. When you select the view, the list of available fields from that view appears as a list of elements that we can use as a facet. For demonstration purposes, we'll select Content Type as the basis of the facet that will be displayed to the site visitor and will give the facet a meaningful name (Content Type), as it will appear as the name of the block in the block layout interface (see Figure 6-30).

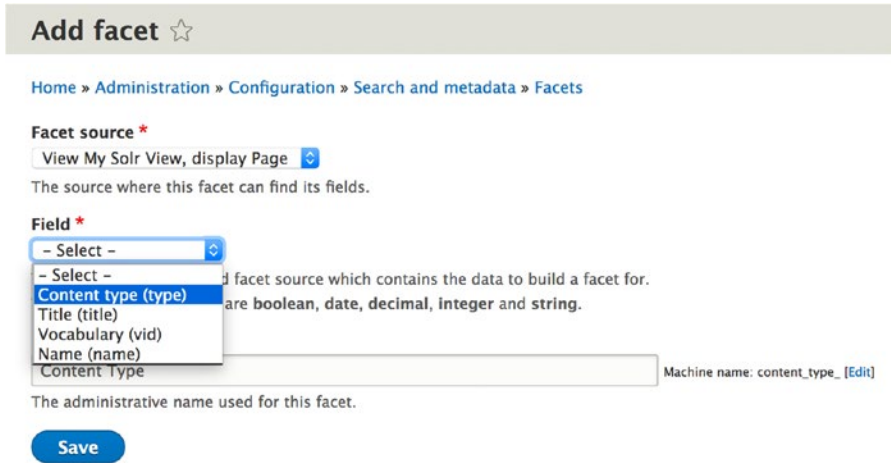


Figure 6-30. Adding a new facet

After saving the facet, navigate to Structure ► Block layout and add the new block named Content Type to the sidebar first region (see Figure 6-31). Set visibility to this block to only appear on the page where your view appears.

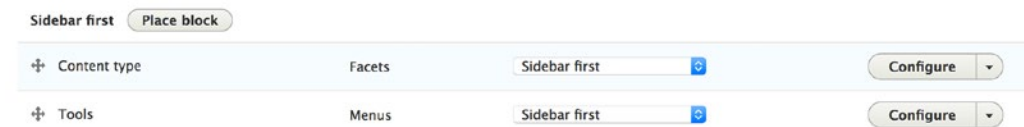


Figure 6-31. Adding the Content Type block to the Sidebar first

After adding the block to the page, navigate to the page that is generated by the view and you'll see the facets that were created (see Figure 6-32).



Figure 6-32. *The facet appearing on the page*

You can create facets for any field that is visible through the view. This is a powerful way for site visitors to drill down into the content that they are interested in.

Federated Solr Search

Using OpenSolr you have the ability for multiple sites to contribute content to a single index, providing cross site searching capabilities as well as the ability to aggregate content across sites and expose that aggregate content through views. This provides an alternative to content deployment across sites as the content in every site is populated in the search index. This is a relatively simple approach for aggregating content and searching across your entire enterprise.

To enable this capability, simply create a new index on a local Drupal site and use the same Solr server and index. You may add as many sites as desired to this single index.

You may also create multiple indexes and selectively add content from specific sites to a specific index, for example, you may have a Drupal site where all product related content resides. Instead of deploying that content across all web sites you may choose to create a “Product Index.” All sites needing product information would then link to that index and utilize the content that resides in that product specific index.

There are many other powerful Solr features. I suggest visiting the Apache web site as well as sites such as OpenSolr’s web site to see the full breadth of capabilities.

Multilingual Support

We live in a world where cultural and country boundaries, while still important, are blurred by the Internet’s capability to connect two people who are geographically thousands of miles apart and enable them to communicate through text, voice, and video. The visitors who come to our web sites may be our next-door neighbors or they may live half a world away. Catering to those who live beyond our region and do not share our native tongue is now more commonplace than ever. Web site designers who break through the language barriers on their sites may attract audiences that they never dreamed of having in the past, and Drupal 8 makes that possibility a reality through its built-in multilingual capabilities.

Getting Started with Multilingual Support

The first step in creating a web site with multilingual support is to determine which languages you want to support. Drupal 8 provides the capability to render your site in nearly any language spoken on the planet. Drupal does not do the actual translation of the content; rather, it facilitates the translation by providing the mechanisms that enable visitors to select which language they want to see (from the list that you offer), and then rendering content that has been previously translated by humans into that language.

After you determine the list of languages that you want to support, the next step is to enable the multilingual modules that are part of Drupal 8 core. Visit the module administration page by clicking the Manage link in the admin menu at the top of the page, followed by the Extend link in the secondary menu. Scroll down the page until you see the list of multilingual modules that are part of Drupal 8 (see Figure 6-33).



MULTILINGUAL	
NAME	DESCRIPTION
<input type="checkbox"/> Configuration Translation	Provides a translation interface for configuration.
<input type="checkbox"/> Content Translation	Allows users to translate content entities.
<input type="checkbox"/> Interface Translation	Translates the built-in user interface.
<input type="checkbox"/> Language	Allows users to configure languages and apply them to content.

Figure 6-33. List of multilingual modules

Configuration Translation provides the ability to translate elements of your site such as the site name, vocabularies, menus, blocks, and other configuration related text on your site. The Content Translation module handles all of the content-related text, such as articles. The Interface Translation module provides an easy-to-use interface for translating elements of your site that are static strings, such as form labels. The Language module enables the definition of which languages your site supports.

Check all of the modules in the Multilingual category and then click the Save Configuration button.

Configuring Multilingual Capabilities

The next step in the process is to configure the multilingual capabilities of Drupal 8. Start by navigating to the Configuration page. Click the Manage link in the admin menu, followed by the Configuration link in the secondary menu. On the Configuration page, scroll down until you see the Regional and Language section (see Figure 6-34).

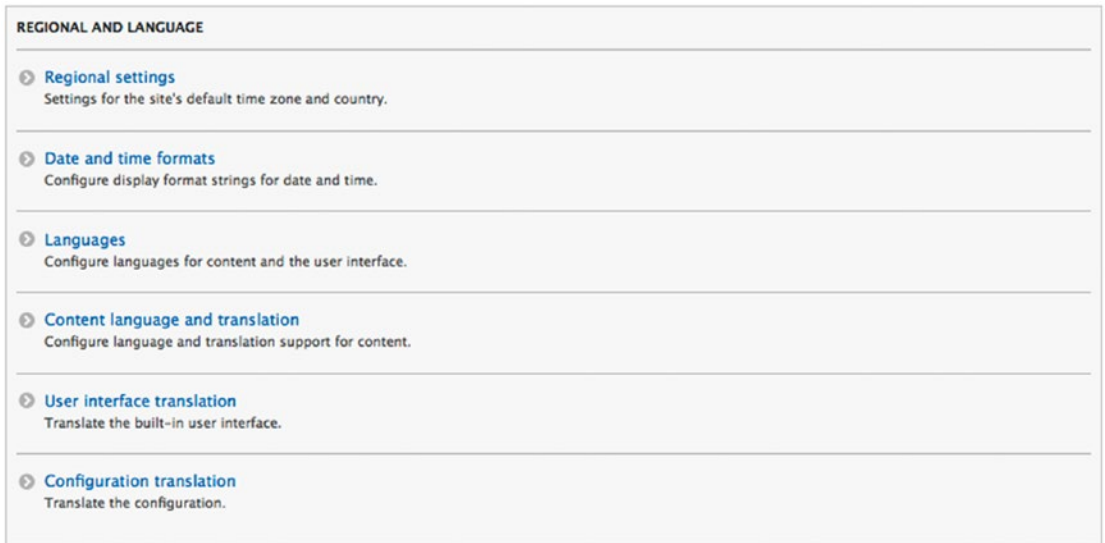


Figure 6-34. Multilingual configuration options

Specifying the Languages

To set the languages that your site will support, click the Languages link on the Configuration page in the Regional and Language section. If you installed your Drupal 8 instance using English as the default language, your Languages page should look like Figure 6-35.

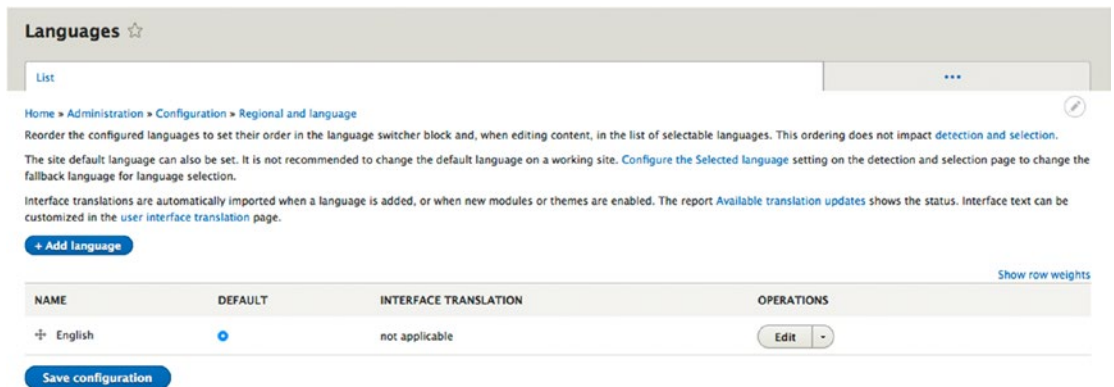


Figure 6-35. Base language

To enable a new language, click the Add Language button and select a language to add to your site from the drop-down list of available languages. Then click the Add Language button (see Figure 6-36).



Figure 6-36. Adding a language

Configuring Language Activation

After setting the list of languages that you want to support, the next step is to specify under what conditions Drupal should switch to a different language. At the top of the Languages page, click the Detection and Selection tab to see a list of options to specify when language switching is to occur (see Figure 6-37).

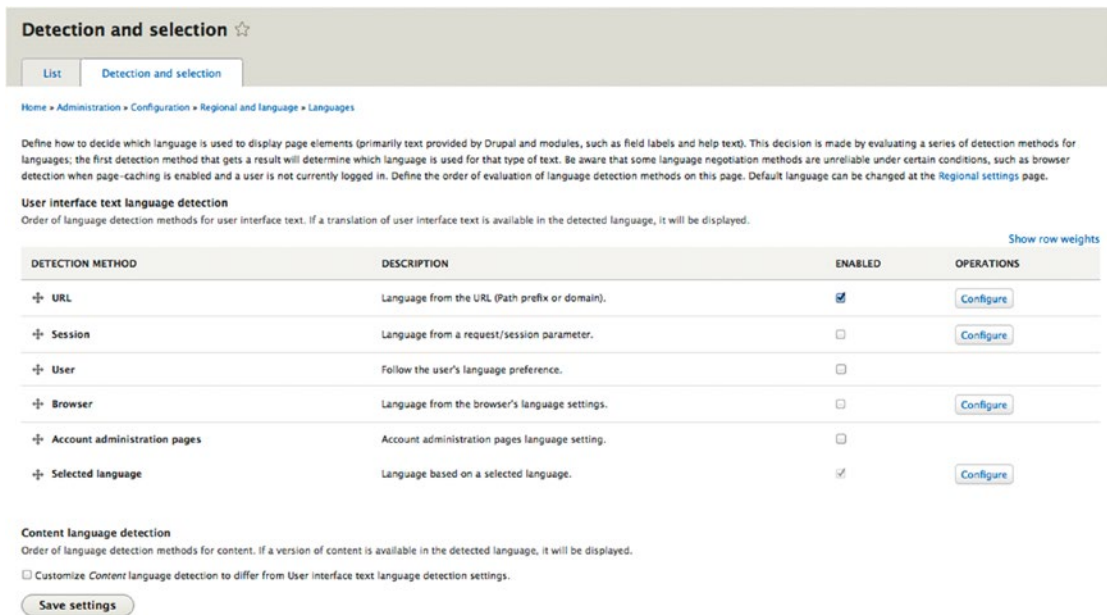


Figure 6-37. Language detection and selection

As shown in the Detection Method column, you have several options for specifying how Drupal decides which language to use to display page elements:

- Specify specific URL patterns that apply to languages, such as <http://example.com/en> for the English version and <http://example.com/ru> for the Russian version.
- Session parameters that are set by custom code and stored in a session variable.
- A user's language preference as set on his user profile.
- The browser's default language settings as set in the user's browser preferences.

- Account administration pages allow you to set a different language for the administrative interface and the content portion of your site.
- A user selecting a language from a drop-down list or radio buttons in a block on your site. Checking this option enables a block that provides the ability to select the visitors preferred language.

For demonstration purposes, check the URL and Selected Languages options and click the Save Settings button to continue.

Some of the options, such as URL settings, provide the ability to configure the parameters that define how those setting will take effect. Click on the Configure button to see the parameters.

By selecting the Selected Languages option, we now have access to a block that provides the ability for users to select which language they prefer. To place that block on a page, navigate to the Block Layout page (Manage ► Structure ► Block layout) and you'll see in the Place Blocks list, under the System category, a block named Language Switcher. Click the Language Switcher link and assign the block to a region provided by your theme. If you are using Bartik, a good choice would be one of the two Sidebar regions. After you select the region, don't forget to click the Save Blocks button at the bottom of the Block Layout page. After enabling the Language Switcher block, your page should look similar to Figure 6-38.

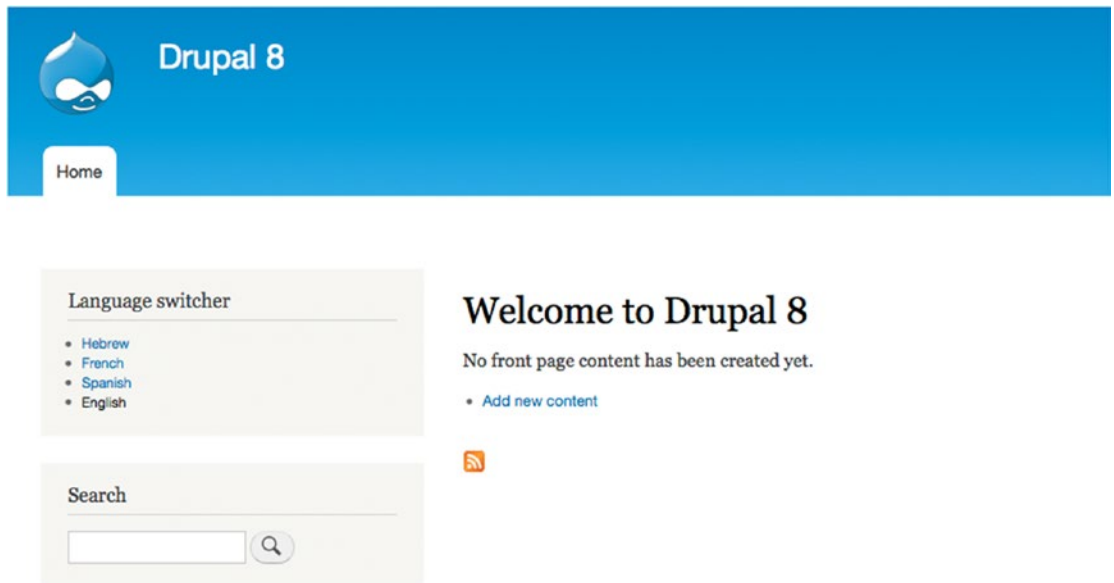


Figure 6-38. Language switcher block

Content Translation Example

With the Language Switcher block in place, you are now ready to take the next steps of translating content. Return to the Configuration page by clicking the Manage link in the admin menu, followed by clicking the Configuration link on the secondary menu. Click the Languages link on the Configuration page to return to the Languages page. After enabling the languages you want to support, you'll see entries for each in a column titled Interface Translation (see Figure 6-39). For each language, this column shows the number of elements that are already translated (the first number) and total number of elements available to translate, where *elements* are field labels, error messages, or other text strings that are defined in template files and modules. As you can see from Figure 6-39, many elements have already been translated by the Drupal community.

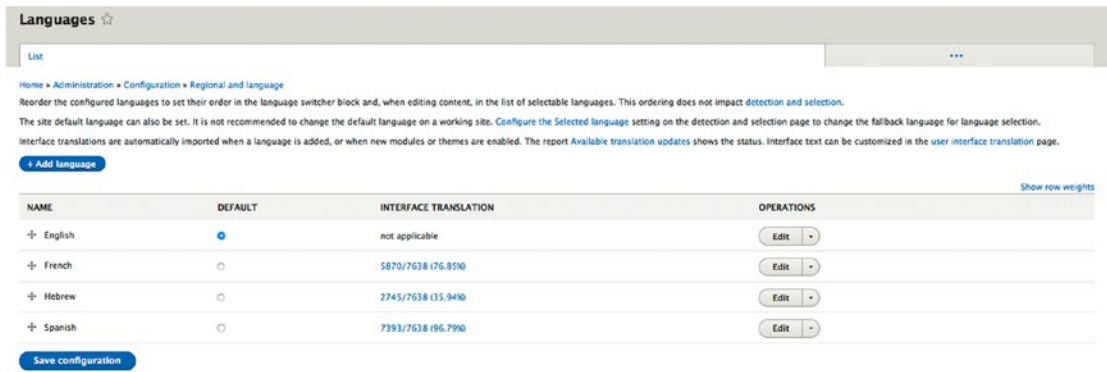


Figure 6-39. Interface translation

Clicking any of the values in the Interface Translation column displays the list of elements, with a text box next to each element where the person doing the translation enters the translated text for that text string (see Figure 6-40). To filter the list to only show elements that do not have a translation, click the Search In list in the Filter Translatable Strings section of the page and select Only Untranslated Strings. Click the Filter button to see the list of items that are missing a translation.

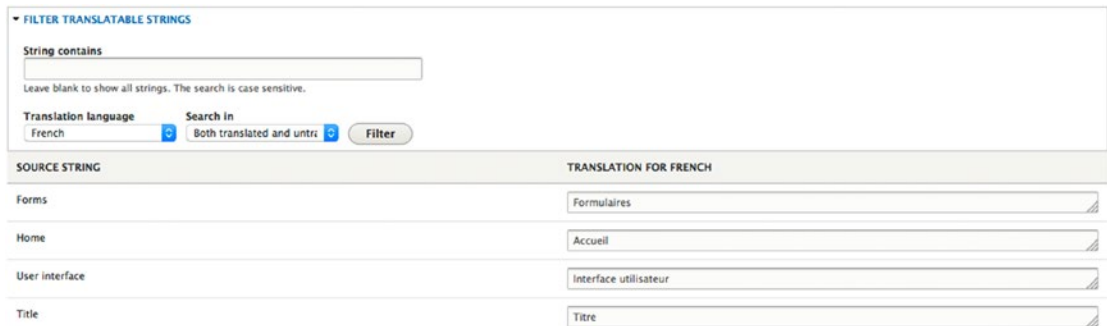


Figure 6-40. Translation of source strings to alternative language

After entering values for some or all of the source strings, click the Save Translations button. Back on the Languages page, the number of strings you have translated will appear in the Interface Translation column, along with the total number of strings and the percentage of strings that have been translated for that language. The total number of strings to be translated may increase as you install new modules, create new forms, or create other features that have interface elements that are translatable. Check this page often to ensure that everything has an associated translation.

Configuring Entities

The next step in the setup of multilingual support on your site is to specify which content types, taxonomy vocabularies, user profiles, or other supported elements are translatable. Return to the Configuration page and click the Content Language and Translation link in the Regional and Language section (refer to Figure 6-34). On this page, you will see a list of checkboxes related to the types of elements on your site that support translation. Simply check the box next to the elements for which you want to provide translation

capabilities. For demonstration purposes, check the boxes for Content, Custom Menu Link, and Taxonomy Term. As you check each box, a list of options appears where you can set the translation capabilities for that element (see Figure 6-41).

Custom language settings

- Comment
- Content
- Custom block
- Custom menu link
- File
- Shortcut link
- Taxonomy term
- User

Content

TRANSLATABLE	CONTENT TYPE	CONFIGURATION
<input type="checkbox"/>	Article	<p>Default language</p> <p>Site's default language (English) <input type="button" value="v"/></p> <p>Explanation of the language options is found on the languages list page.</p> <p><input type="checkbox"/> Show language selector on create and edit pages</p>
<input type="checkbox"/>	Basic page	<p>Default language</p> <p>Site's default language (English) <input type="button" value="v"/></p> <p>Explanation of the language options is found on the languages list page.</p> <p><input type="checkbox"/> Show language selector on create and edit pages</p>

Custom menu link

TRANSLATABLE	CUSTOM MENU LINK	CONFIGURATION
<input type="checkbox"/>	Custom menu link	<p>Default language</p> <p>Site's default language (English) <input type="button" value="v"/></p> <p>Explanation of the language options is found on the languages list page.</p> <p><input type="checkbox"/> Show language selector on create and edit pages</p>

Taxonomy term

TRANSLATABLE	VOCABULARY	CONFIGURATION
<input type="checkbox"/>	Tags	<p>Default language</p> <p>Site's default language (English) <input type="button" value="v"/></p> <p>Explanation of the language options is found on the languages list page.</p> <p><input type="checkbox"/> Show language selector on create and edit pages</p>

Figure 6-41. Content language configuration

Checking the box for the Article Content Type, for example, displays additional details as to which elements of that item are translatable. For the Article Content Type, this includes the title, body, comment settings, image, and tag fields. For demonstration purposes, check all the boxes for all the elements, followed by clicking the Save button.

Translating Content

With the pieces in place, the next step is to author content in the site’s native language and translate it to the various languages that your site has been configured to support. For demonstration purposes, assuming you checked the box for Article in the previous step, create a test article in the native language set for your site. Click the Manage link on the admin menu, the Content link in the secondary menu, and the Add Content button on the Content page. Select Article as the type of content to create. Note that a new field appears on the Create Article form, called the Language Select list. For demonstration purposes, select the default language that represents the base language of your site (e.g., if you installed the English version of Drupal 8, select English from the select list). On my Drupal 8 example site, I created an Article in English using “This is a test article” for the title, and “Hello World this is a test article in English” as the body text. Save and publish the Article by clicking the button at the bottom of the form. After saving the article, you’ll notice a new Translate tab at the top of the Article form (while logged in as an administrator with content-editing permissions set). The new tab allows you instant access to the translate feature (see Figure 6-42).

This is a test article

View Edit Delete Translate

Submitted by [admin](#) on Sun, 02/09/2014 - 22:31

Hello World this is a test article in English.

Figure 6-42. Translate option

Clicking the Translate tab displays a list of all the languages you specified while configuring multilingual support and shows the current translation status for each of those languages for the content item that you are working with (see Figure 6-43).

LANGUAGE	TRANSLATION	SOURCE LANGUAGE	STATUS	OPERATIONS
Hebrew	n/a	n/a	Not translated	Add
French	n/a	n/a	Not translated	Add
Spanish	n/a	n/a	Not translated	Add
English (Original language)	This is a test article	n/a	Published	Edit

Figure 6-43. Language translation status

Clicking the Add button for a specific language brings up the node edit form for that piece of content, allowing you (or another human translator) to see the original-language version of that content item, with the ability to override that version with the translated version. Pick one of your languages from the list and give it a try. Here is my test article being translated into French (see Figure 6-44).

