django cms Documentation

Release 4.1.0

django CMS Association and contributors

Dec 28, 2023

CONTENTS

Note: This is a new section in the django CMS documentation, and a priority for the project. If you'd like to contribute to it, we'd love to hear from you - join us on our friendly Slack group.

This is a user guide for django CMS. The intended audience for this guide are content creators and site administrators.

Django CMS sites are highly customisable and varied. The examples here follow the quickstart project from Github. This provides certain features, which are entirely optional for your own site, and the features provided in the quickstart are used in this guide to demonstrate how django CMS can be used.

The origin of this document is a guide provided by Kapt.mobi. It has been translated from the native French and extended.

CHAPTER

ONE

GUIDE FOR CONTENT EDITORS

1.1 Tutorial

Start here as a new django CMS content editor:

- Getting to know the user interface
- Understanding the page tree
- Creating content

1.2 How-to-guides

Practical **step-by-step guides** to get things done in the most simple way as a content editor

1.3 Concepts

What's a placeholder? What's a plugin? Understand key concepts of django CMS.

1.4 Reference for content editors

Reference material for installed plugins.

1.4.1 Tutorial

Note: This is a new section in the django CMS documentation, and a priority for the project. If you'd like to contribute to it, we'd love to hear from you - join us on our friendly Slack group.

It's strongly recommended that you follow this tutorial step-by-step. It has been designed to introduce you to the system in a methodical way, and each step builds on the previous one.

Logging-in

Once you have created your site with the django CMS components you will want to customise the site and add content. To this you will need to log in and edit it.

There are different ways of accessing your user account in django CMS. Please refer to what the respective developer of your project has told you.

The default way is to add `/?toolbar_on` to the end of your URL and use the toolbar to log in. The banner or login page will appear, asking you for your username and password. When you first log in, these two elements have been set by the developers. Therefore, if you are not sure how to fill in these fields, please contact them.

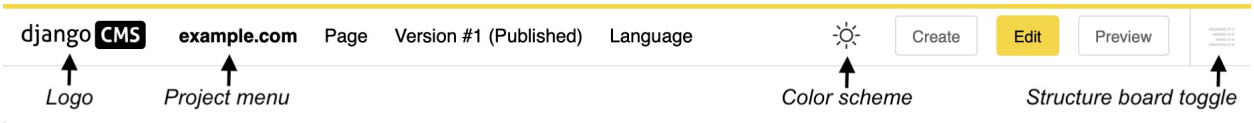
Logging in will allow you to access your site to add, modify or delete content. You will be able to perform a lot of actions using the django CMS menu bar, which is present when you are logged in to your site.

Upon logging into your site, you will be able to see your site's pages and other content with a toolbar at the top of the visible area. This toolbar is used to make changes to your site.

The toolbar

Overview

When connected to your site, the Django CMS toolbar appears. It can also be called a menu bar since it offers you several menus performing various actions.



Depending on the modules present on your site and the page you are on, you can find various elements. The only ones that will always be present are "Project", "Page" and "Language". The other proposals that may appear depend on the CMS modules present on your website and where you are located. For example, "Blog" will only be displayed if you have this module and you are on your blog page or article pages.

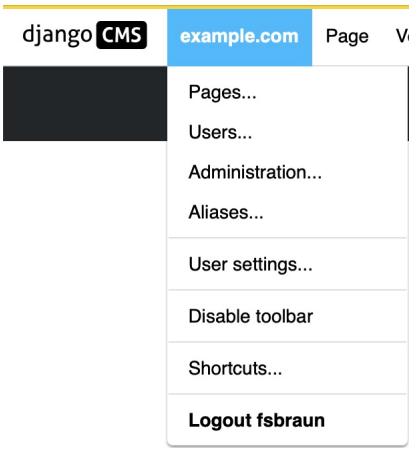
Another element that is constantly present in your toolbar: the django CMS logo. If you click on it, it will simply take you back to your home page.

To the right of the logo follows the project menu, the page menu and potentially some more menus: Here the version and language menu. The language menu will be missing in single-language configurations.

On the right handside you see the color scheme toggle which you can use to switch the color scheme of django CMS interface elements between light and dark. It is followed by action buttons which are highly context dependent. Finally, on the right hand corner, there is the structure board toggle which might be disabled. It is central to editing content.

The Project menu

This is the part of the django CMS toolbar that corresponds to your site. Normally, the name of your website is displayed instead of "example.com".



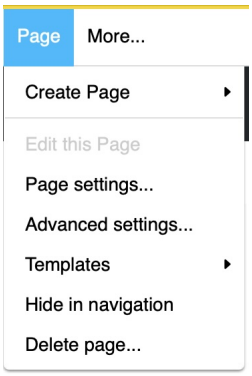
Note: You might not see all of the elements described here, or even more elements. This strongly depends on your django CMS setup.

- **Pages…** - allows you to access the page tree of your site by opening the side bar.
- **Users…** - gives you access to the user management panel in the side bar.
- **Administration…** - Allows you to manage various features via the administration window.
- **Aliases…** - Allows you to access content elements which are re-used within your sites, such as footers or announcements.
- **User settings** - Allows you to change the language of the administrator interface and the menu/toolbar.
- **Disable Toolbar** - allows you to completely disable the toolbar and the front-end editor, regardless of the login and status of the person logged in.
- **Shortcuts** - gives access to your shortcuts.
- **Logout <username>** - the user will be logged out.

Generally, by choosing one of the items of this menu, the content will be displayed in the sidebar overlay or in a new page. In the case of the sidebar, you can close it with the cross at the top right.

Note: If you disable the toolbar, you can only make it reappear by adding `?toolbar_on` to the end of the URL in your browser window.

The "Page" menu



The "Page" menu allows you to perform the following actions:

- **Create new pages** - you have the choice to create a new page, a new sub-page or to duplicate the page you are on.

  Note: A parent page is a page that contains a group of several other pages of which your page is part.

  A sibling page is a page of the same level as your current page: they belong to the same group. Sibling pages either have both no parents or the same parent page.

  A sub page is a page nested within the parent page group. A parent page is the "parent" of the sub page which if also called child page.

  This is called page inheritance.and explained in more detail in the section on the tree structure of your site later in this document.

- **Edit this page** - allows you to edit the page you are on. Changes to a page are only possible if your are in edit mode.
- **Page settings…** - gives you access to the settings of the current page.
- **Advanced settings** - allows you to access the advanced settings of the page. Amongst the advanced settings of a page are their permissions.
- **Templates** - allows you to choose the template of your page. You can also choose to have your page inherit the template of the "closest parent", i.e. apply the template of its parent or sister page.
- **Hide/show in navigation** - toggles if the page should appear in the sites navigation
- **Delete page** - deletes the page.

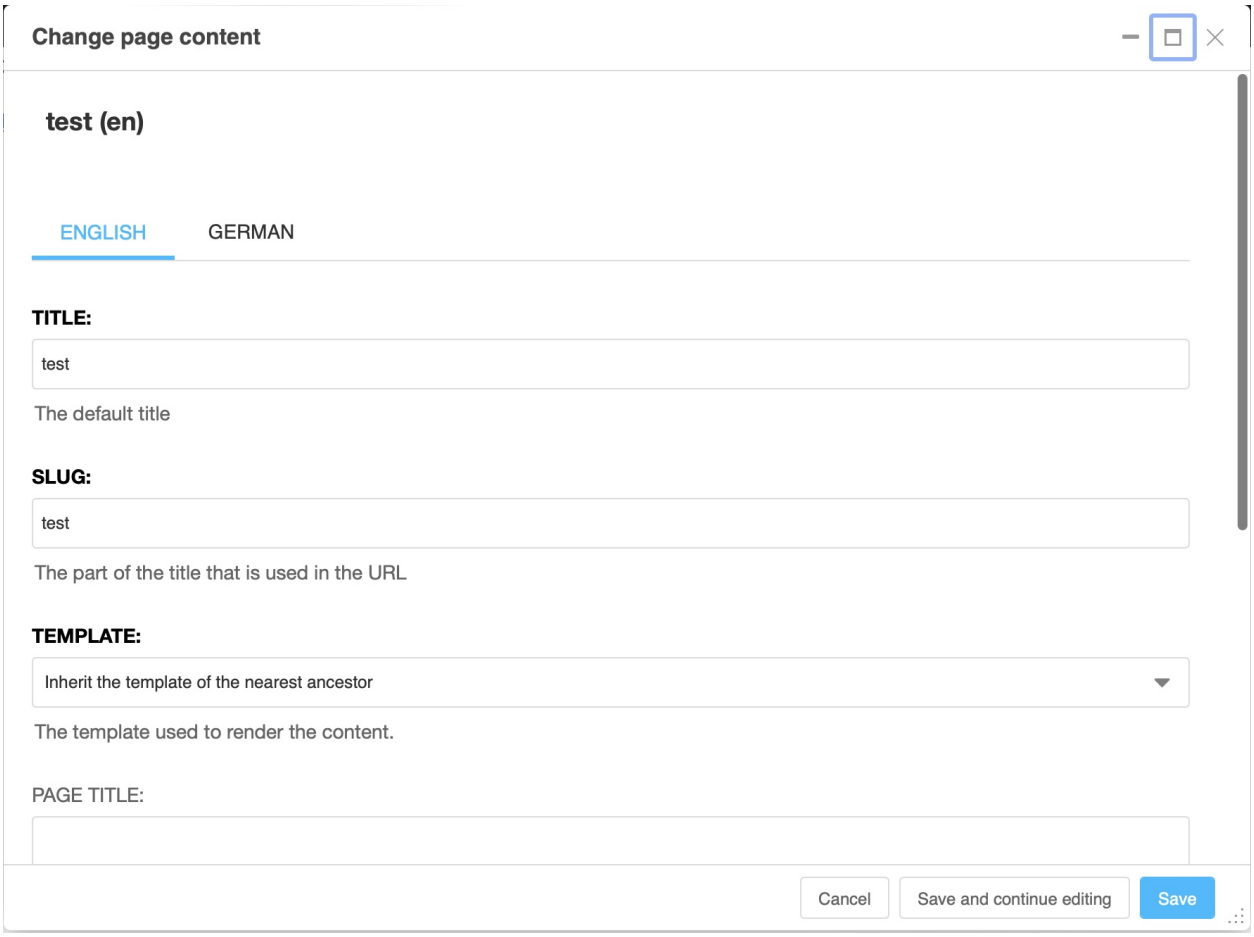
Page settings

The screenshot below give you a quick overview of the basic settings dialogs for your page.

We will see them in more detail when we create your page, but here are some important information:

- **Title:** field dedicated to the title of your page that you enter when creating your page.
- **Slug:** automatically generated, it is what appears at the end of the URL.

You will probably not use the advanced settings because they are mainly used for redirects.



Note that the page settings are available separately for all languages of the page. Use the language tabs to navigate between settings in different languages.

The version menu

The version menu allows you to manage current and past versions of the page you are viewing. The menu title includes the version number of the current page (counted by language) and its status: * Draft: The copy you are currently editing * Published: The version that currently is publicly visible. * Unpublished: Any version that once was publicly visible but now is not any more. * Archived: Any draft version that has been archived for later use.

You can perform the following actions:

- **Manage versions…** - get a list of all versions of this page in the sidebar.
- **Compare to version <x>…** - get a visual comparison on how the currently viewed version differs from any other version of the pasge. Difference are highlighted by colors either on the page or the page's source code.
- **Discard Changes** - makes all changes of the current draft undone

The language menu

The "Language" menu allows you to switch between the different language versions on the page you are viewing, and to manage the different translations.

You can add a missing translation, delete an existing one or copy all plugins and their content from an existing translation to the current one.

The action buttons

On the right hand side are the action buttons.

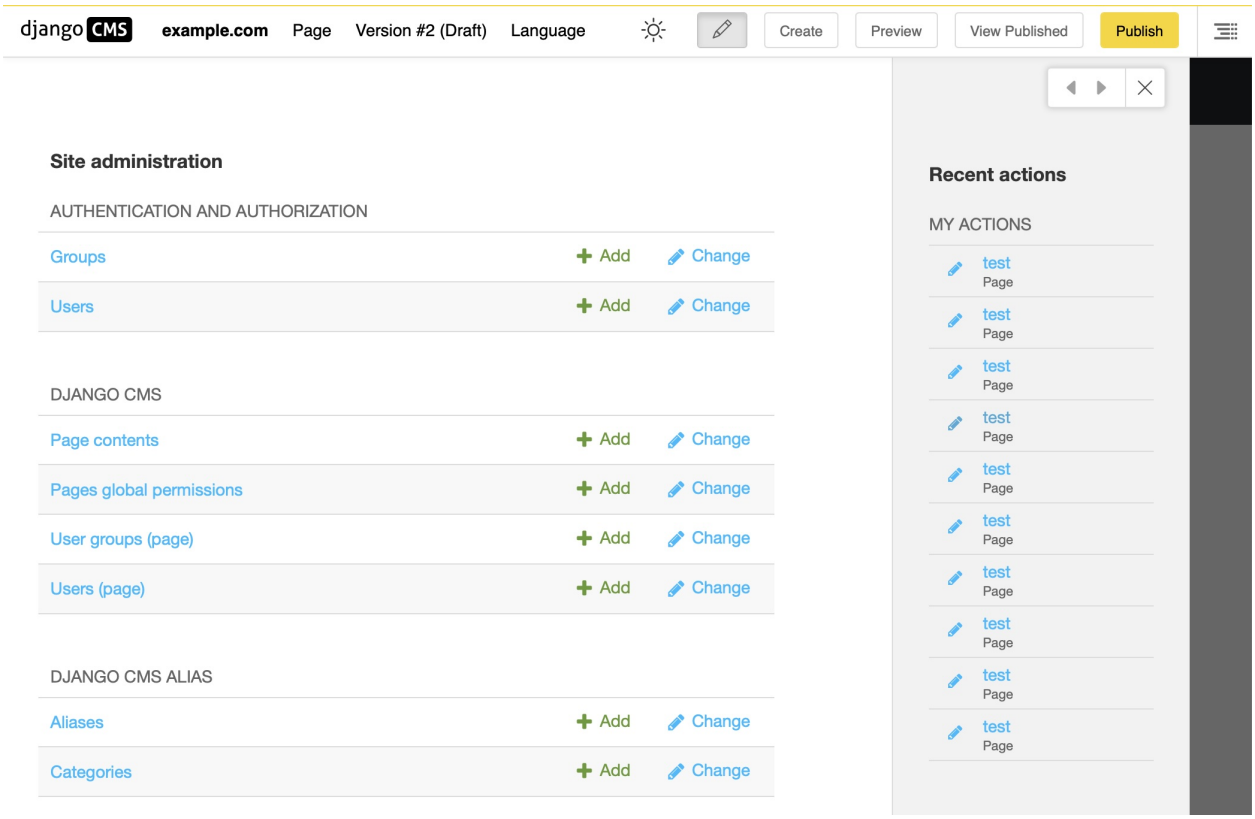
- The **Create** button opens up the content creation wizard which allows you to easily create a new page or potentially other content elements installed at your site.
- The **Edit** button to open the page to make changes in editing mode, (appearance of the structure menu & possibility to double-click to modify the content),
- The **New Draft** button to create a new draft copy of the page end open it in edit mode.
- The **Preview** button switches to the preview mode. Here, no changes are possible. Also works for non-public content, such as unpublished pages.
- The **View published** button redirects to the currently visible version of the page.
- The **Publish** button which makes the current draft the new published content. A previously published version of the page becomes unpublished.

Administration side bar

Introduction

When you click on "Administration" in the project menu, the administration or admin window opens. It covers most the content of your page. If you want to go back to your page, just click the close button or into the area outside the side bar. It covers the full administration interface of your site, including those added by django CMS. Depending on the modules installed on your site, its content may vary. It is identical to the content on the sites admin pages.

From this page, you can access the different administration sections of your site: management of pages, media files, users, ...



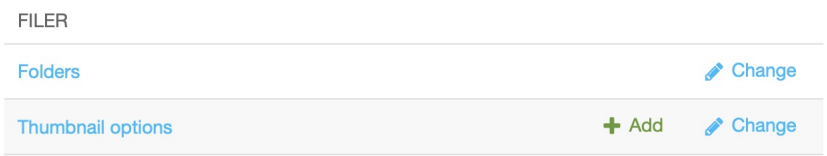
The media library

The Media library allows you to access all your files, folders and images on your site. You reach the media library through the Administration menu entry in the project menu. It opens in the sidebar.

By accessing it, you will be able to modify the titles, alt tags, captions and other information of your images, and also manage your folders and files.

Access the Media library

To manage the files on your site, look for the Filer section and click on “Folders”:



Description of the Media library

Media files live in folders just like in a regular file system.

The greyed out "Unsorted uploads" file contains all the images that do not belong to any folder and that have been directly uploaded to a CMS page or a blog post.

The blue folder "Blog" might contain all your picture folders used in your blog. You will be able to manage all the photo folders, add or delete them…

Warning: Important: Before deleting a folder or an image, make sure that the contents of the folder or the image are not used on your site. If, for example, you delete an image while it is present in an article, it will disappear from this same article.

Warning: Would you like to see more content here? Please join us on Slack and the Slack `#workgroup-documentation` to add content here.

The page tree structure

The tree structure of your site allows you to access all your pages. Many features and information are present in this part of your site. To better understand them, we suggest you look at how to access your tree structure, and understand the principles that are linked to it and its features.

See the page tree

To access the tree structure of your site, go to the project menu (which has the name of your site, or "example.com" on the screenshots here), and choose "Pages…". The side bar opens and shows the complete tree structure of your site. That is to say, all the pages that make up your site: visible and non-visible, pages and sub-pages, published or not.

Alternatively, you can reach the page tree by clicking "Page contents" in the admin sidebar, which you find in the "Django CMS" section.

Principles of the parent, child and sibling pages

This section is intended to give you a better understanding of what is meant by page inheritance and the names given to it.

The page tree structure is divided into different sections: these are the pages at different levels. Each page can be nested in another one, i.e. a page can contain several sub-pages. This page is then called the "parent page" and its sub-pages are its "child pages". A child page can also have its own child pages. The idea is to organize your site in several different levels so that it is clearer for the user.

To better understand, let us take the image of a tree as a metaphor:

- Your **home page** is the trunk, it presents all the thematic pages of your site.
- The **top-level pages** are the branches of your tree and have other small branches. We call these top-level pages "parent pages".
- The small branches correspond to the **child pages** of your top-level pages (the "parents"). Each parent page may or may not have child pages, just as the branch of a tree may or may not have smaller branches.

- The leaves of your tree represent the content of your pages or the sub-pages of your child pages, which do not have any child pages of their own.

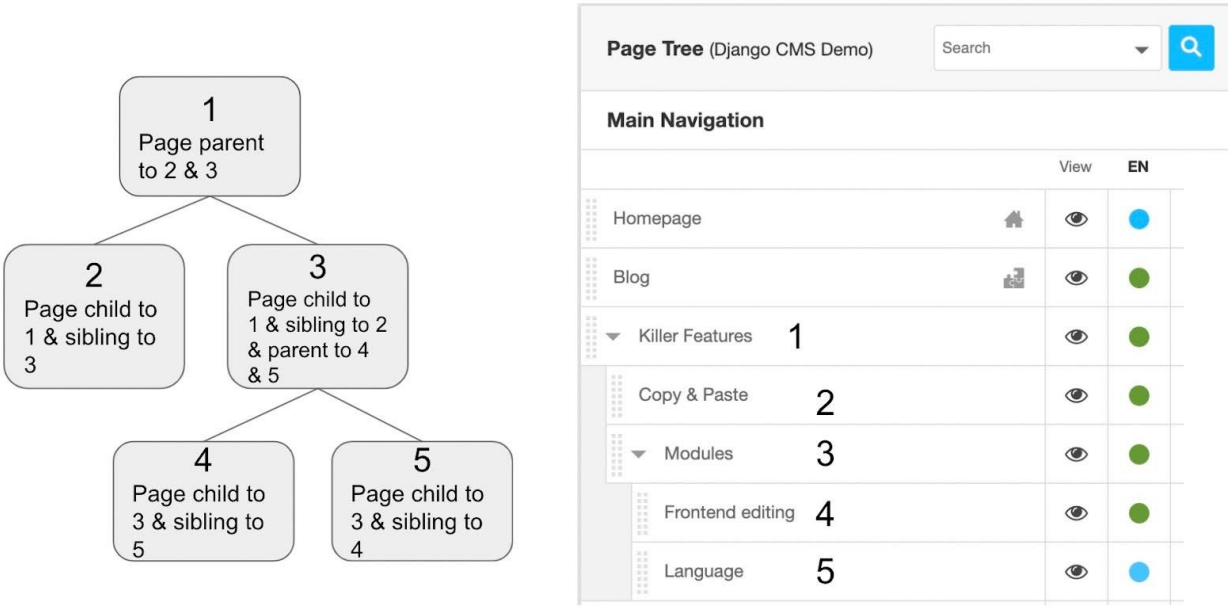


Fig. 1: The page tree structure: Related pages, child and sibling pages. A small branch is composed of pages 3, 4 and 5. The leaves are in this example are pages 2, 4 and 5.

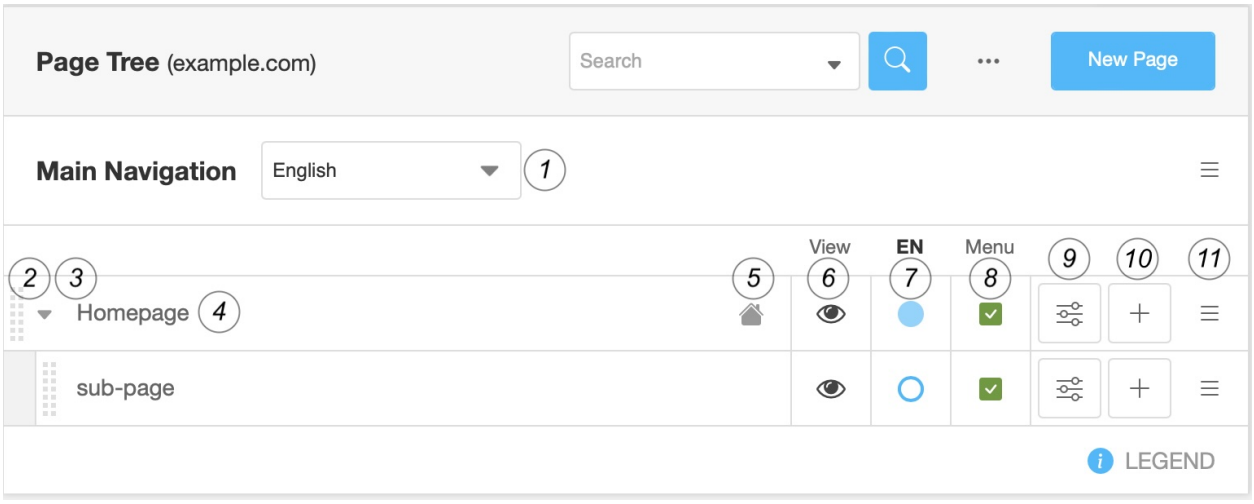
Each page is thus nested in the others when they have a common point: all your blog posts correspond to the parent page “Blog” for example. All your pages having the same the same central goal can be grouped together in a group and form the set of child pages of a parent page bearing the name of this topic. Thus, your parent page allows access to the content of your daughter pages and helps the user to better find his way around your site in relation to what he is looking for and wants to do.

As for a sibling page, it is simply a page at the same level as those around it in the tree. In a group of child pages, all pages are siblings. The same applies to parent pages.

Managing the page tree

The page tree is a list of all the pages of your site. It gives you an overview of your pages and their status.

By default, the following information is displayed:

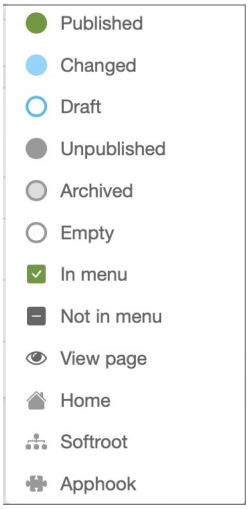


From left to right, you will find the following items:

1. The **language menu** which decides which language version of the page tree you see
2. The dotted bar on the left to **move the pages** using Drag & Drop
3. The **hide arrow**, present only in the case of a parent page. It shows or hides the page's child pages.
4. The **title of the page** in the selected language. If this says "Empty title" there is no version of that page in the selected language, yet.
5. The **root indicator** (house) indicating which page is presented at the root URL of the site. A puzzle symbol at this place indicates that the page is handled by a third-party application.
6. The **eye icon** allows you to preview the page
7. **The publication status menu menu. Its color provides the following information:**
    - **Blank**: the translation of the page into this language does not exist.
    - **White with blue border**: An unpublished draft of the page exists. There is not public version of the page.
    - **Green**: the translation exists and the page is published,
    - **Blue**: changes on the page have not been published. If you go to this page, the button at the top right will suggest you to publish the changes so that they are visible to the user.
8. **"Menu" indicates whether the page appears in the navigation menu of your site or not,**
    - Green / check: the page is visible in the navigation menu
    - Gray / unchecked: the page does not appear
9. The **settings button** opens the page settings.
10. The **add button** adds a child page.
11. The **hamburger context menu** offers additional actions: * Copy * Cut * Paste * Delete… * Set as home * Advanced settings * Miscellaneous information: the date of the last change, the access (restricted or not), the author of the last modifications.

At the top right is a small menu, composed of a search button allowing you to find a specific page in your tree, a "…" button to choose the site whose tree you want to see (in many installations you will only see one site), and a button to add a new page.

At the bottom right there is the legend with a complete list of all symbols used in the page tree view. Open it by clicking on the information icon:



Creating places for content

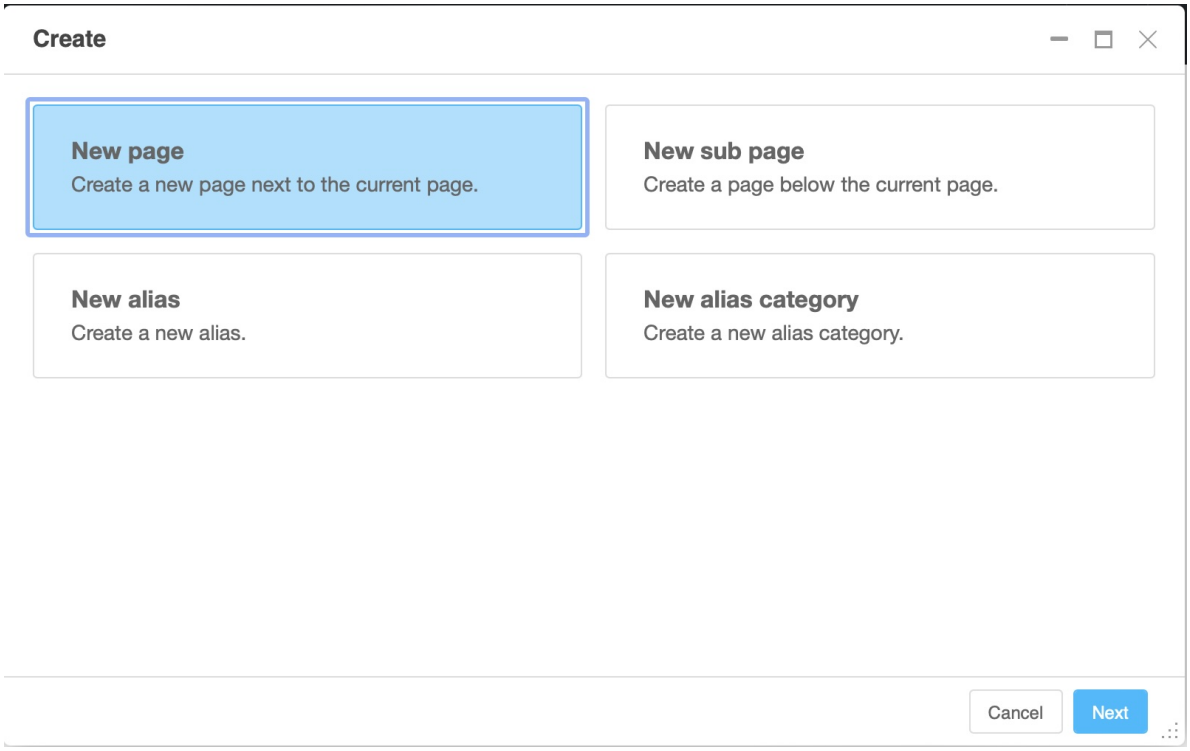
Content needs a place to live. Typically those are pages, but you also may have other content models, like aliases (chunks of content that are re-used elsewhere) or blog posts.

We will work with pages in this guide.

Create a page

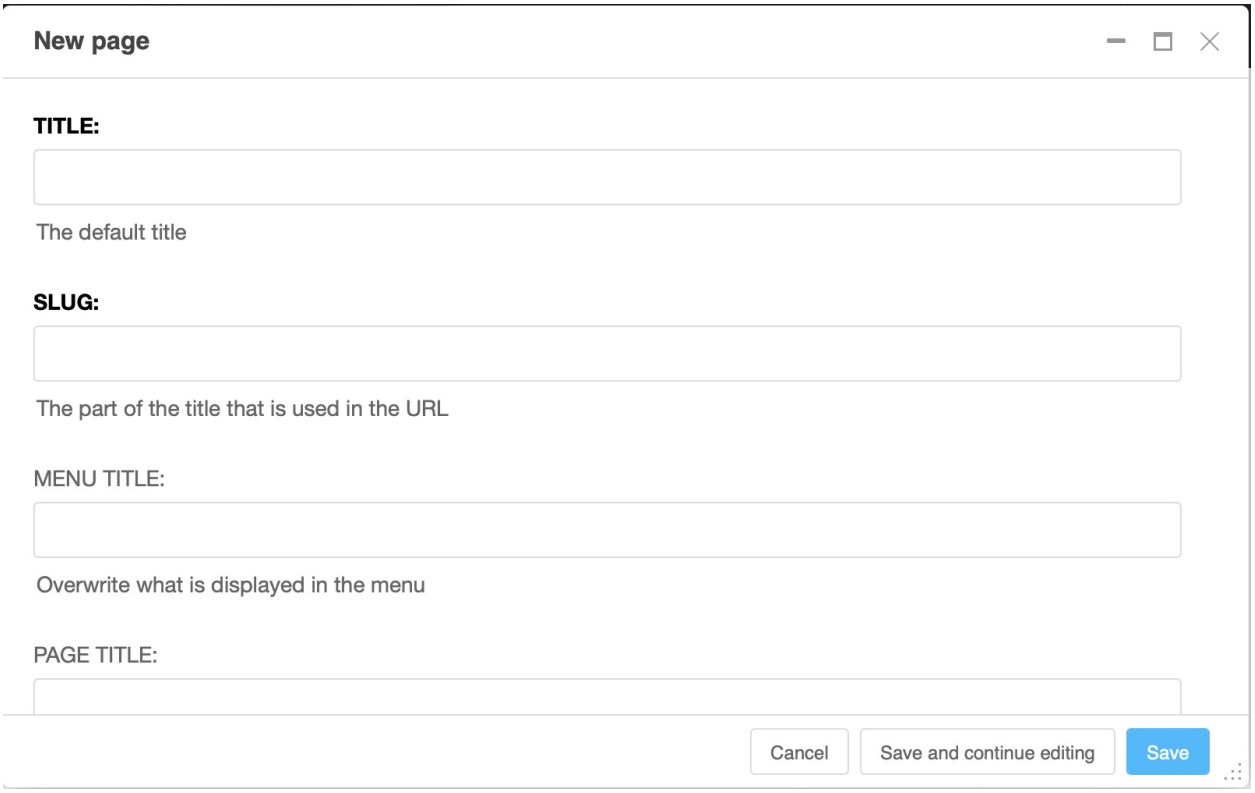
To create a new page, you have three options:

1. Go to the **project menu**, select "Pages…". The sidebar will appear. Click on the button "New page" to open the page dialog box.
2. Use the **wizard** by clicking on the "Create" button at the top right of the toolbar. The wizard dialog appears, where you can select "New page" or "New sub page"



3. In the **"Page" menu**, select "Create a page" then "New page…" (or "New sub-page…). The page dialog box appears.

The page dialog is more extensive than the wizard dialog and contains all elements of the page settings.



In the page dialog box, give a title and possibly a menu and page title then save. The slug field will be filled automatically

based on the page title. Of course, you can manually change it.

Your newly created page is displayed, as well as the Django CMS menu/toolbar with the main content management tools. A newly created page is empty. Adding content is done via the structure board, in the upper right corner.

The color, the size of the text… everything is generated automatically according to the page template of your site which in turn should be based on your individual graphics design. To change the design, there are options to modify the page template. If you want to change the design, talk to the developers to review it.

Change page settings

To view the page settings, go to "Page Settings…" in the Page menu.

These are the same parameters that are displayed when you create a page using the page dialog box. You can also access them by selecting the settings icon that appears in the page tree.

Required fields

- **Title**: it will be used by the template of your site and displayed at the top of the page, in the tab of the site on the browser, and in the favorites. Search engines will also use it. It is therefore a field that must be filled in.
- **Slug**: this is part of the URL of your page. In general, you will want it to reflect the title of your page, and it will be automatically generated from it in an appropriate format. Keep in mind that it is always valuable to keep your slugs as short and sweet / pleasant / concise as possible.

Note: For your SEO, it is essential that the title and slug of your page contain words related to the content of your page.

Optional fields

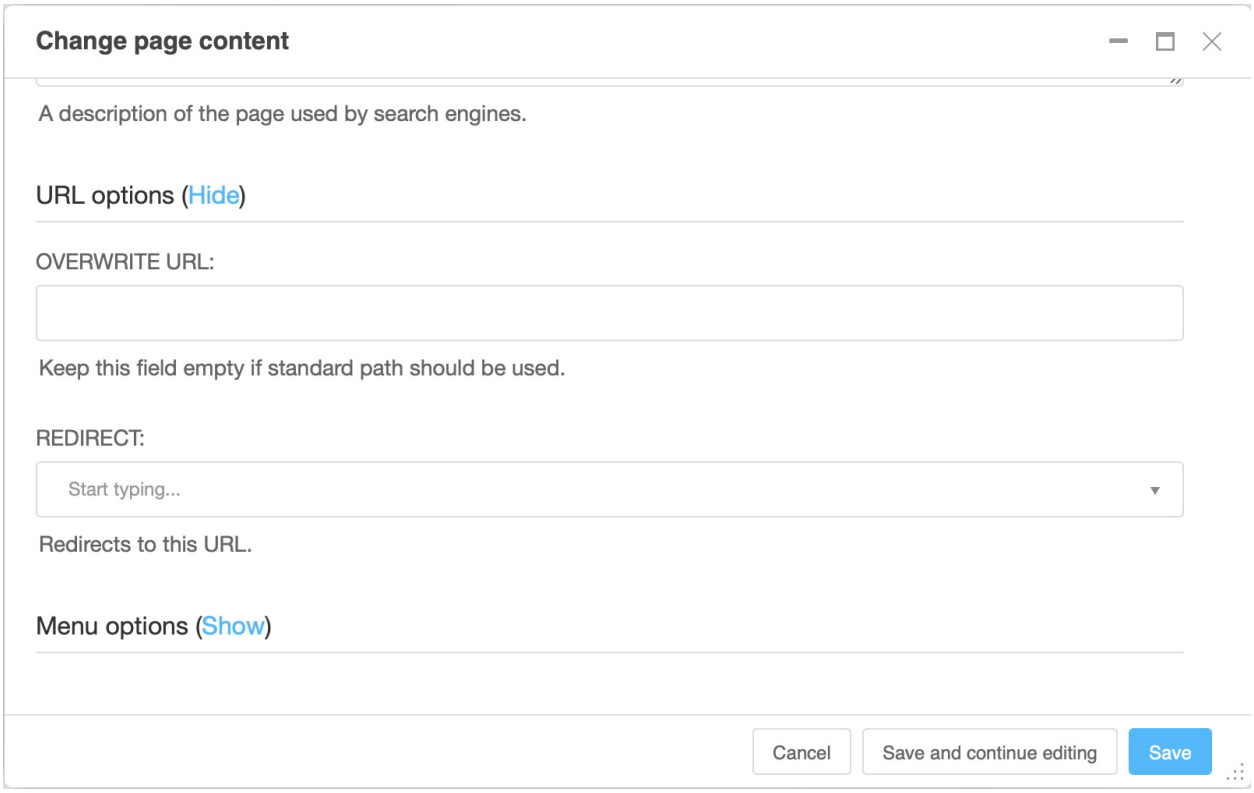
- **Menu title**: If set this is used in your site's navigation menu otherwise the title field of the page is used. This useful if the entire title is too long to be used in navigation. For example, the title "Partner Company:Our Story" will be far too long to work well in the navigation menu, especially for your users on smartphones. "Our Story" would make a more appropriate menu title.
- **Page title**: If set it replaces the title which is displayed in search engine results and in the browser tab.

Brief review of the different title fields

1. The **title** field is mandatory, it is the default title that is displayed on your page. It contains the keywords and is reused for menus and as page title of those fields do not overwrite it.
2. The **menu title** is only used in menus where titles should be shorter.
3. The **page title** appears in the search engine results and in the browser tab. It is longer, and contains a maximum of keywords. Be careful, however, not to go too far beyond the 60 characters, which seems to be the current display limit on Google.

URL options

The URL options appear as a section a bit down in the page settings dialog. You might need to click "show" to make them visible.



- **Overwrite URL**: allows you to modify the URL. By default, the URL of the page is the slug of your current page, placed after the slug of its parent pages. For example, its default URL could be: `/a-propos/company/our-vision/`. Overwriting the URL would allow you to shorten it to `/our-vision/`, even though the page still keeps its parents and children organized in the same way (About page, Company daughter page, Our Vision grandchild page). We do not recommend using this function.
- **Redirect**: Allows you to redirect the user to a different page. This is useful if you've moved content to another page but don't want to break the URL that your users may have bookmarked or affect the rank of that page in search engine results.

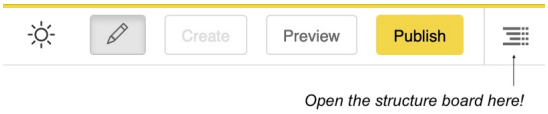
Filling in content

Now that you have created a place where your content can live, you can start filling in actual content. Like the page tree, content is structured in so called "plugins" which live in plugin trees inside placeholders.

Structure board

Your page will have one or more placeholders to fill with content. They are displayed in the **structure board**.

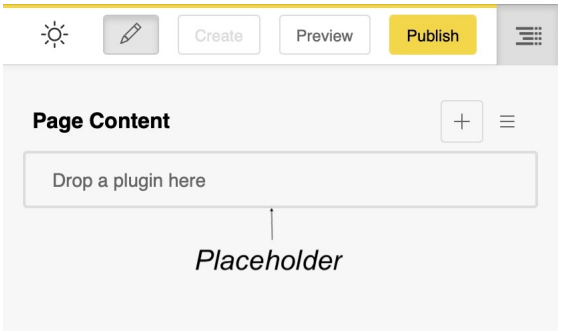
To add plugins to your page (or alias, or blog), you need go to open the structure board by clicking the button on the far right side of the menu bar.



There are different types of plugins for adding content, each with a specific purpose. The ones you will mainly need are the "classic" rich text, image, video, link and button plugins.

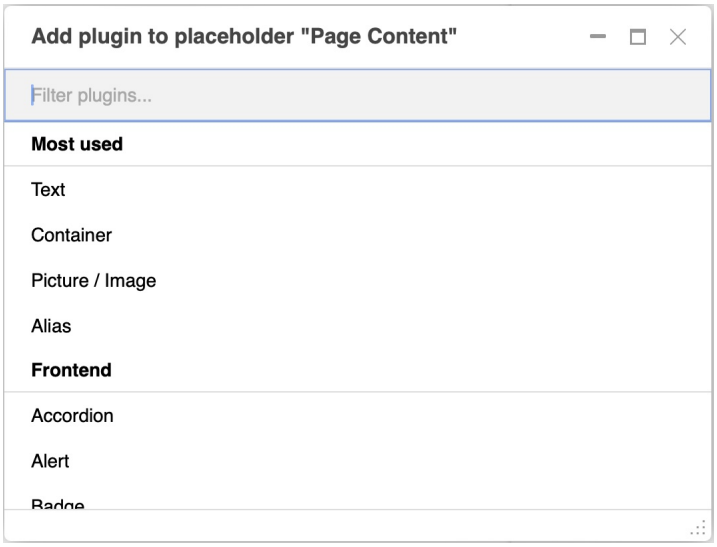
Note: Which plugins are available on your site strongly depends on the installation. Take this guide as a blueprint on how to interact with plugins. Even if you have different plugin options installed the editing process is always the same.

Once you have clicked on the structure board button, the structure board will open on the right side of your browser window. It will show one or more placeholders.

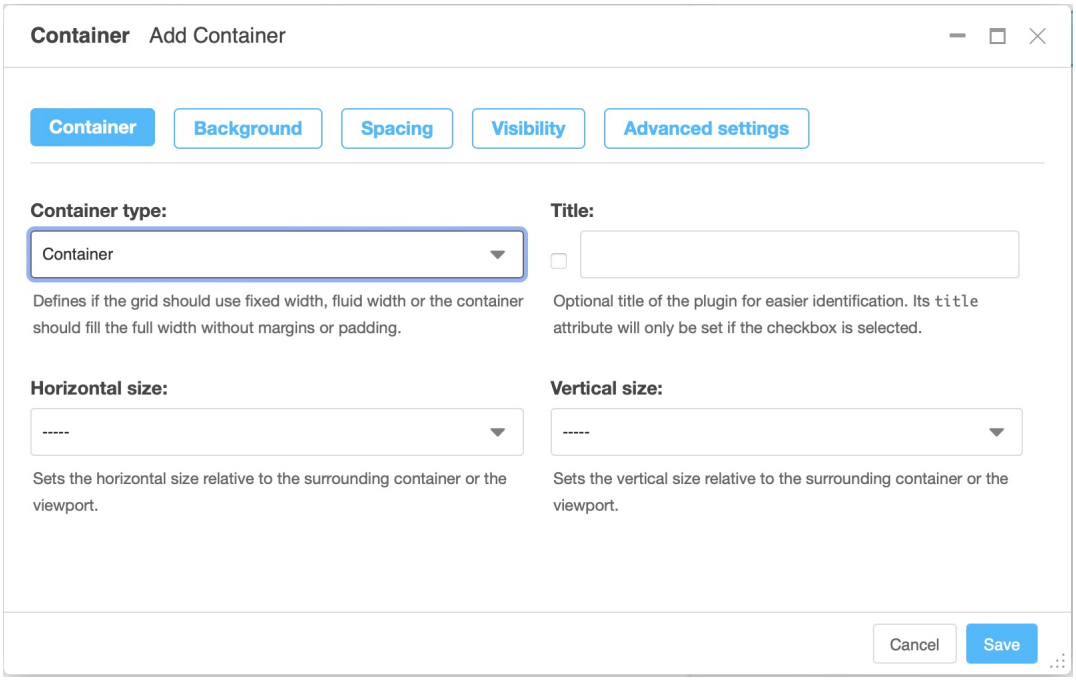


Adding plugins

To add content to a page, click on the button . A dialog box will open, showing you the list of content you can add. Make your choice, a new dialog box opens.



Fill in the fields (in the container example below you do not have to fill any fields - all are optional), and save. The new content is displayed on your page.



Note: The exact content of the add or edit plugin depends on the plugin itself. The container plugin in the example is one of many plugins provided by the django CMS Frontend package which are designed to structure your page.

Those plugins do have a set of tabs (colored in blue) at the top of the dialog and offer a wide set of design options, most of which are optional.

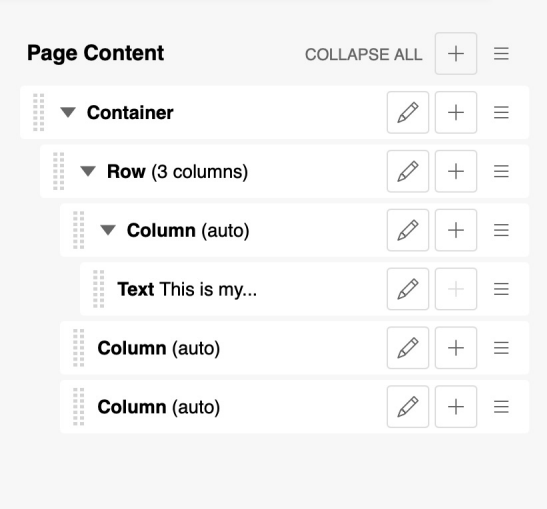
Repeat the operation as many times as you want and for as much content as you want.

If you want to move content to arrange their layout, for example to move an image before or after text, use the "drag & drop" function of the CMS via the notch on the left of the content type.

You can also add content to change your layout; some of this content is nestable (you can put text in columns, which

are themselves inserted in a section).

If this element allows the addition of nestable content, the add plugin button will be available at the same level as the module title. The triangle icon will appear next to the dotted line of the drag and drop button to show or hide the nested content:



Tip: If you **hover** over the structure menu items while pressing the **SHIFT** key, the content displayed on the CMS page will be highlighted, so you can see what each plugin corresponds to.

Editing plugins

You can change existing plugins by either

- **double clicking** on the content in the page (when in edit mode)
- **clicking the pencil icon** of the plugin in the structure board.

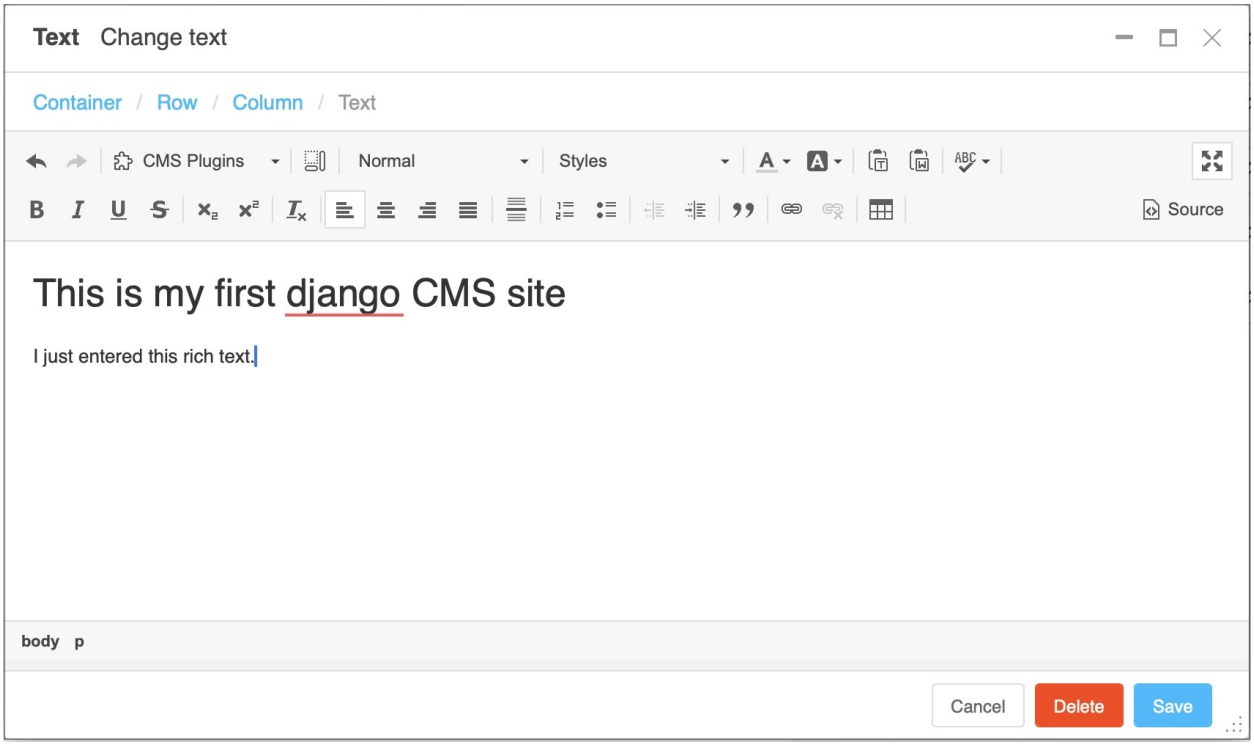
Editing works exactly like creating a plugin

Integrating content

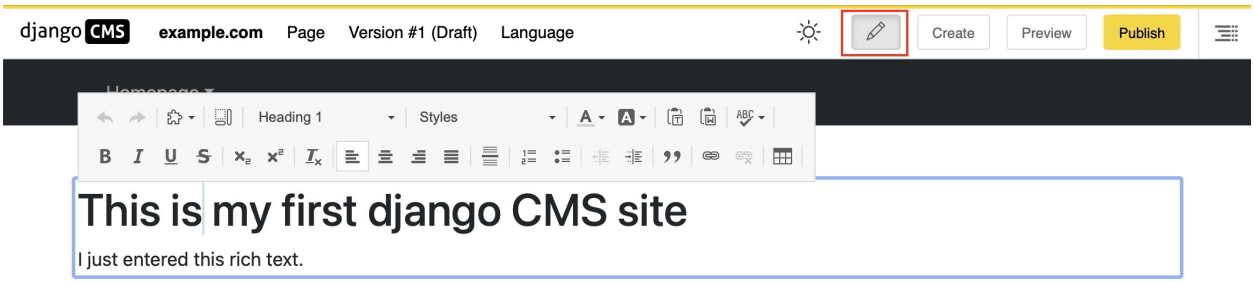
When you add content to your page, you may want to integrate text, a video, clickable buttons or links into your text.

Text

django CMS includes a rich text editor. Its interface is particularly simple, since it only consists of the text you want to enter:



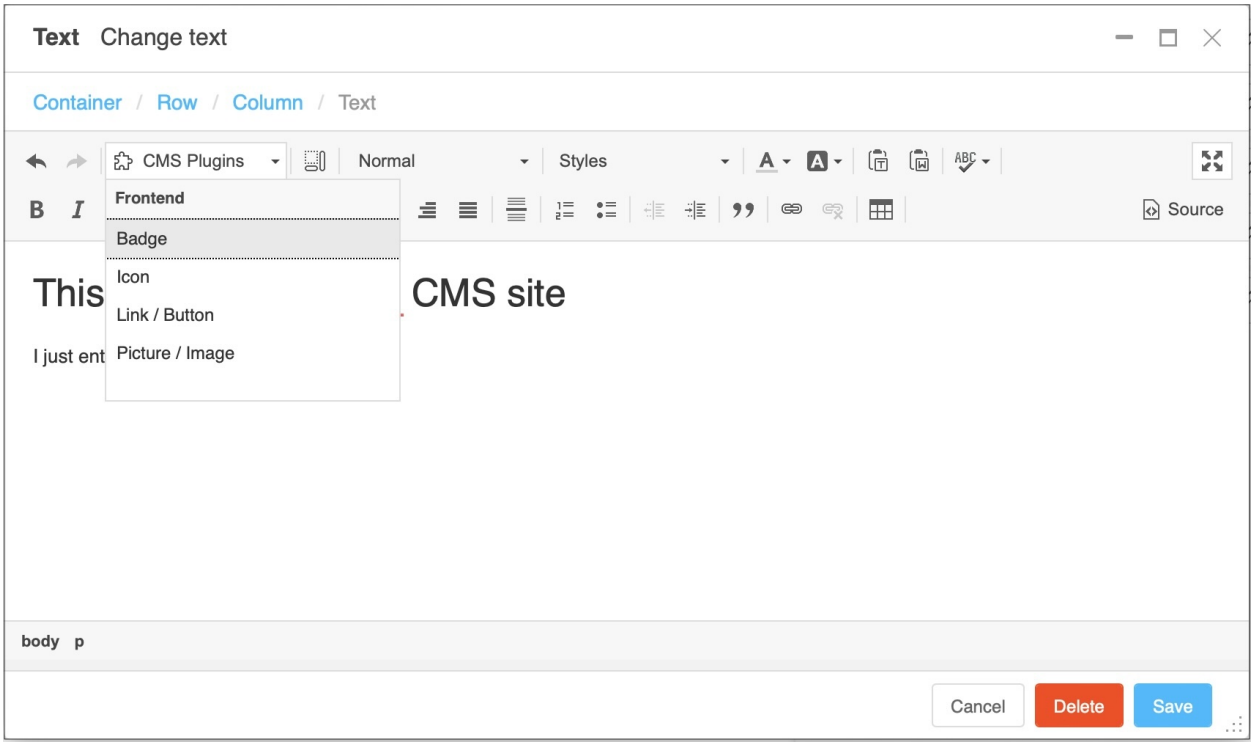
Tip: If your installation has inline editing enabled, you can even edit text right on the web page if the pencil button in the toolbar is activated:



Tip: All plugins show a few words of summary in the plugin tree. Those plugin trees can get huge, though. To keep an overview, use a feature of the Container plugin:

The title field is a text field to briefly describe the container content. It will be displayed in the plugin tree. It is a good practice to use separate containers for different sections of your page and fill the title for quick navigation in the plugin tree.

Some plugins can even be added directly to a text plugin using the "CMS Plugins" menu within the text editor. This is useful for adding dynamic links to your text. Dynamic links are links to other pages of your site which - should the destination change its URL - will be automatically kept up-to-date.



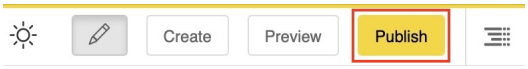
Warning: Would you like to see more content here, e.g. on images, videos, links and buttons? Please join us on Slack and the Slack `#workgroup-documentation` to add content here.

Publishing content

Note: This section only applies to installations such as the django-cms-quickstart project which use the djangocms-versioning package for versioning. django CMS' modular architecture allows for other versioning packages, too.

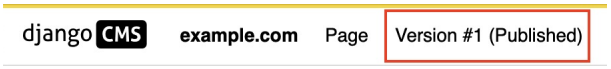
Once you have finished filling in your newly created page (or blog post), it is saved as a draft. It will not be published until you decide to do so. As an editor, you can view drafts, but any site visitors will only see published pages.

To publish a page, click on the highlighted "Publish" button in the toolbar. This button is visible in the page's edit mode. (If there is a highlighted "Edit" button instead, click it first to get to edit mode.)



When published you will be taken to the published page or the "manage versions" dialog, depending on the setup of your site. If you're taken to the manage versions dialog, click on the eye button of the top listed version to get back to the page you just published.

You will see in the toolbar that the version menu now shows that the content is "published".



Version states

Since django CMS 4, pages and other publishable content can have more than the two states "draft" and "published".

Note: Versioning is managed by an optional package like django CMS Versioning. Your installation might manage versions differently. This guide assumes that django CMS Versioning is installed.

The **"published" state** marks the only version of the page that is currently visible to web site visitors.

The **"draft" state** of a page is the one which the editor can change.

Any version that has been published at any point in time will get the **"unpublished" state** once it is either unpublished manually or implicitly when another version of the page is published.

Versions in the **"archived" state** have not been published but retained by a user for later use.

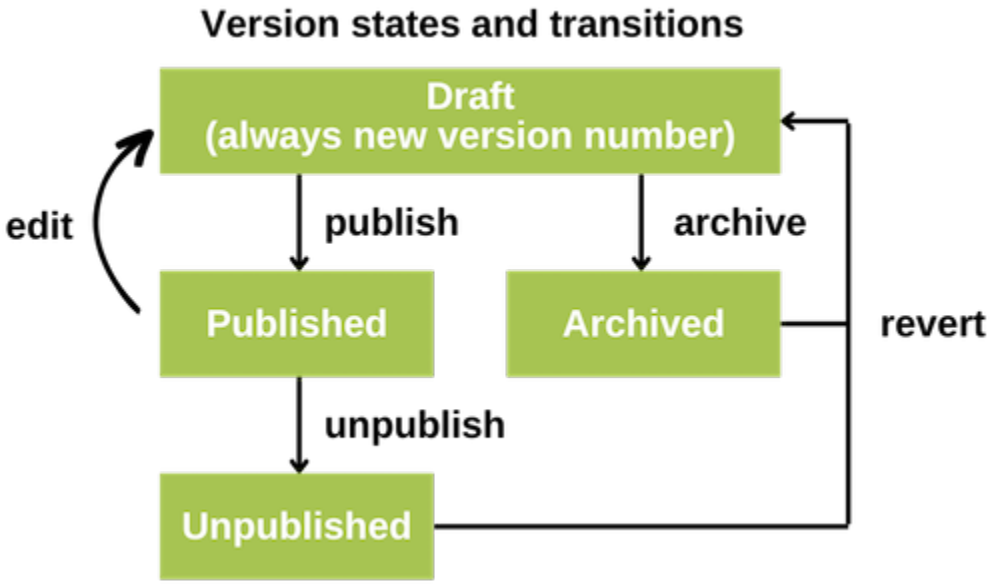
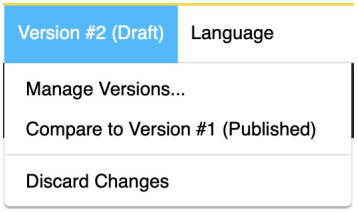


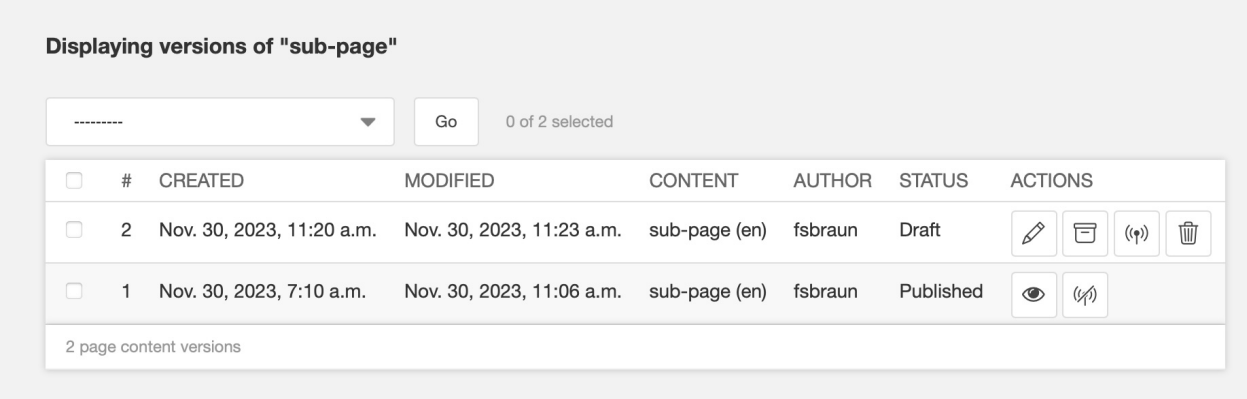
Fig. 2: This diagram summarizes the possible states and the actions which create a change in the state of a version.

Managing versions

To manage versions of the currently displayed page, go to the Version menu and select "Manage Versions…":



This opens the sidebar with the manage versions dialog:



You will see (a potentially long) list of all versions created, ever. On the right side of the list you see the action buttons. You can

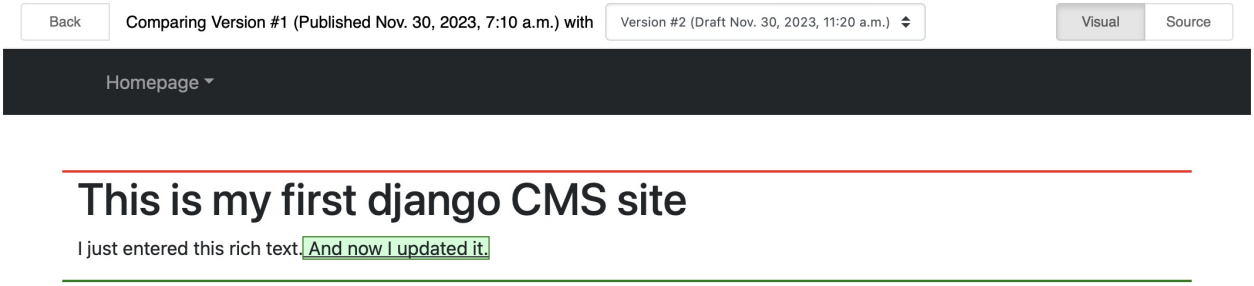
- **Pencil**: Go back to the **edit** view of a draft
- **Archive**: Archive a draft version for later reuse.
- **Antenna**: Publish a draft
- **Litter bin**: Delete a draft
- **Eye**: View an old version. Note: Those cannot be edited any more.
- **No-antenna**: Unpublish a currently public version
- **Undo**: Revert an archived or unpublished version into a new draft version

Comparing versions

With many versions being available, it sometimes is difficult to keep track of changes. You can visually compare two versions by

- Either selecting two versions in the manage versions dialog and the choosing the "Compare versions" from the actions dropdown and hitting "Go".
- From the Version menu

An example of a visual comparison is shown here:



Added content is marked green, deleted content is marked red.

Tip: If you want to see the changes in the generate HTML code you can select "Source" in the upper right of the compare view.

Note: Touch-screen users

django CMS supports touch-screen interfaces, though there are currently some limitations in support. You will be able to complete the tutorial using a touch-screen device, but please consult *Using touch-screen devices with django CMS*, and see the notes on *Device support*.

1.4.2 Concepts

Note: This is a new section in the django CMS documentation, and a priority for the project. If you'd like to contribute to it, we'd love to hear from you - join us on our friendly Slack group.

django CMS uses some specific concepts. This section explains those concepts from a content editor perspective.

Placeholders

In Django CMS, placeholders are special markers within templates that define regions where content can be edited and managed by content editors. These placeholders act as slots or containers within a template where various types of content can be added, modified, and rearranged through django CMS' frontend editor and its structure board.

Here's how placeholders work:

1. **Defined Areas in a page:** Designers identify specific areas within their HTML templates where content can be dynamically inserted. Editors can access these defined placeholders and add or edit content directly through the user-friendly frontend editing interface without needing to touch the underlying code. Pages can be rendered using different templates.
2. **Content Manipulation:** Editors interact with placeholders to add various types of content elements or plugins (such as text, images, videos, forms, etc.) to these designated areas. They can modify existing content, rearrange elements, or remove content as needed.
3. **Flexibility and Customization:** Django CMS allows for the creation of custom plugins that can be inserted into placeholders. These plugins offer a wide range of functionalities and content types, giving content editors the flexibility to create diverse and engaging web pages without requiring developer intervention for each content update.

Ultimately, placeholders in Django CMS facilitate a clear separation between the presentation layer (templates) and the content, enabling content editors to manage and update website content easily without needing extensive technical knowledge or modifying the underlying code.

Plugins

Django CMS plugins are modular components that represent content. In Django CMS, content components are organized into placeholders within templates. These placeholders can be filled with various plugins to add diverse content components.

Django CMS plugins offer content editors an intuitive way to manage and enrich the content of web pages without needing technical expertise or diving into code. Here's how they benefit content editors:

1. **Ease of Use:** Content editors can work inside the frontend editor and use the structure board, where they can see available placeholders on a page. They can easily add, edit, or remove content elements (plugins) within these placeholders using a simple drag-and-drop interface or form-based interactions.
2. **Content Variety:** Plugins provide a wide range of content types that can be inserted into placeholders, such as text, images, videos, galleries, sliders, forms, maps, and more. This variety enables editors to create rich and diverse web pages without relying on developers for every content update.
3. **Customization:** While there's a set of default plugins available, each installation has its own set of plugins activated and might also use custom plugins tailored to specific needs. These custom plugins can encapsulate complex functionalities or unique design elements that content editors might require for their content. Those custom plugins are not within the scope of this user guide.

In essence, Django CMS plugins empower content editors to manage and present content effectively, providing them with the tools to create engaging and dynamic web experiences without needing to delve into the technical intricacies of web development. These plugins empower editors to create and manage content without needing to write code or modify templates each time they want to add specific elements to a webpage.

Publishing

Note: This section only applies to installations such as the django-cms-quickstart project which use the djangocms-versioning package for versioning. django CMS' modular architecture allows for other versioning packages, too.

In Django CMS, publishing and versioning are crucial aspects for editors managing content. Here's a breakdown of how they work:

Publishing

Publishing refers to making content available to website visitors. In Django CMS, you typically create or edit content using a **draft**, which means the unfinished content or changes are not immediately visible to the public. Editors can work on content privately until it's ready for publication.

This is the typical workflow:

1. **Create Draft:** Editors create or modify content only in a draft version within the CMS admin interface.
2. **Preview:** They can preview how the content will appear on the live site before publishing.
3. **Publish:** When ready, editors can publish the changes to make them visible on the live website for visitors to see.

Published content cannot be changed any more. To make additional changes, create a start over the process and create a new draft based on the published version.

Managing versions

Versions in Django CMS keep track of changes made to content over time. This feature allows editors to revert to earlier versions of content if needed and view the history of modifications:

1. **Version History:** Django CMS maintains a history of all changes made to a particular piece of content.
2. **Compare Versions:** Editors can compare different versions of content to see what changes were made between each iteration.
3. **Rollback:** If necessary, editors can revert to a previous version of the content, effectively undoing recent changes.

These functionalities empower editors to manage content effectively, ensuring quality and control over what gets published on the live site while maintaining a history of changes for reference or restoration purposes.

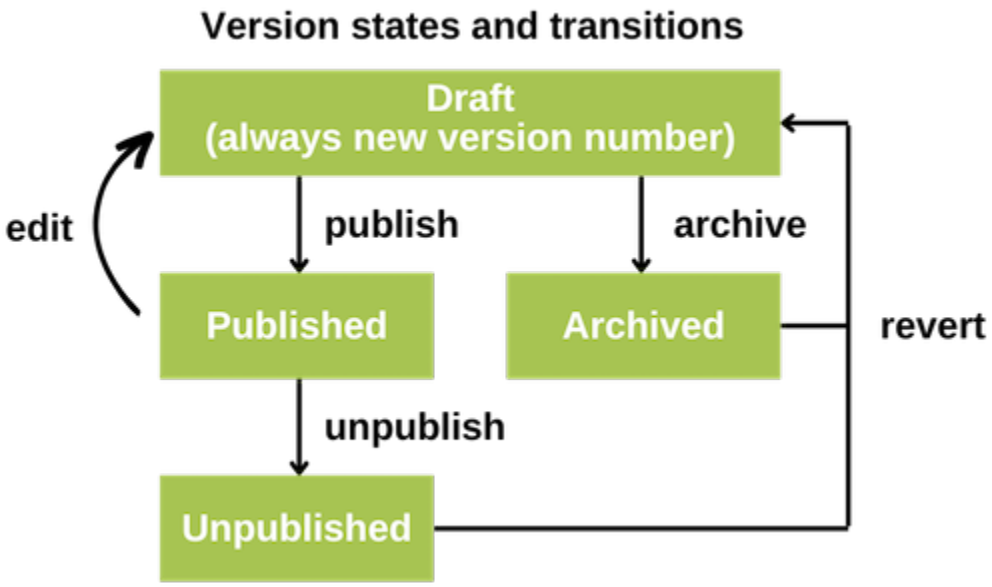


Fig. 3: This diagram summarizes the possible states and the actions which create a change in the state of a version.

Locked versions

Note: This feature is not enabled in all installations of django CMS. See django CMS Versioning documentation for more information.

Version locking in Django CMS is a feature that automatically locks each draft version of a content item to prevent unintended modifications or edits. A locked draft can only be changed by the person who created the draft. This functionality is particularly useful when you want to ensure that not two editors make changes to a specific content, ensuring changes do not interfere with each other.

Aliases

Note: This section only applies to installations such as the django-cms-quickstart project which use the djangocms-alias package.

In Django CMS, the Alias plugin is a powerful tool that enables content editors to display in as many places as they like without duplicating it. Essentially, it creates a reference or link to content that resides in a separate place - called an "Alias".

Key aspects of the Alias plugin include:

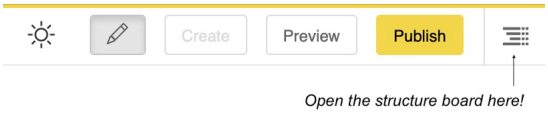
1. **Content Replication:** Rather than copying content, the alias content is referenced either in a page template or by using the Alias plugin.
2. **Cross-Page Content Sharing:** It facilitates sharing content across different pages within the site. For instance, if there's a section that needs to appear in multiple places but should have consistent content, the aliases can be employed to achieve this.
3. **Maintaining Consistency:** Using aliases ensures consistency in content across various parts of the website. If the original content gets updated, all aliases referencing it will reflect those changes instantly.
4. **Saves Effort and Reduces Errors:** Instead of manually replicating content across multiple pages (which could lead to discrepancies or errors), the Alias plugin streamlines the process and reduces the chance of inconsistencies.
5. **Ease of Management:** Content editors can manage content in one place while displaying it in multiple locations, making it easier to maintain and update information without having to navigate through numerous pages.

In essence, aliases in Django CMS serves as a smart reference system, allowing content editors to reuse existing content across the site while maintaining consistency and efficiency in content management.

Structure and content modes

In Django CMS, the concepts of "structure" and "content" modes refer to different aspects of managing and arranging the plugins when editing a page (or other content).

You can toggle between structure and content mode by clicking the button on the far right of the toolbar.



1. **Structure Mode:**
   - **Purpose:** Structure mode is primarily concerned with the arrangement and organization of the different plugins within placeholders.
   - **Functionality:** In structure mode, you can define the overall layout of your pages by creating plugins (regions within a template where content can be placed) and arranging these plugins within the placeholders of your page.
   - **Usage:** This mode is used for setting up the basic plugin structure of a page, determining where various types of content (text, images, videos, etc.) are placed within the layout.
2. **Content Mode:**
   - **Purpose:** Content mode focuses on the actual content that fills the placeholders created in structure mode.

- **Functionality:** When in content mode, you can add, edit, and manage the specific plugins by double-clicking. This includes text, images, videos, widgets, and any other content types supported by Django CMS.
- **Usage:** Content mode is used for changing or populating the placeholders with actual content. Users typically work in content mode when they want to add or modify text, images, or any other elements on the pages.

In summary, structure mode deals with the layout and arrangement of placeholders on a page, while content mode involves filling those placeholders with actual content. These modes work together to provide a structured yet flexible way of managing the overall structure and content of a website within Django CMS.

Using touch-screen devices with django CMS

Important: These notes about touch interface support apply only to the **django CMS admin and editing interfaces**. The visitor-facing published site is **wholly independent** of this, and the responsibility of the site developer.

General

django CMS has made extensive use of double-click functionality, which lacks an exact equivalent in touch-screen interfaces. The touch interface will interpret taps and touches in an intelligent way.

Depending on the context, a tap will be interpreted to mean *open for editing* (that is, the equivalent of a double-click), or to mean *select* (the equivalent of a single click), according to what makes sense in that context.

Similarly, in some contexts similar interactions may *drag* objects, or may *scroll* them, depending on what makes most sense. Sometimes, the two behaviours will be present in the same view, for example in the page list, where certain areas are draggable (for page re-ordering) while other parts of the page can be used for scrolling.

In general, the chosen behaviour is reasonable for a particular object, context or portion of the screen, and in practice is quicker and easier to apprehend simply by using it than it is to explain.

Pop-up help text will refer to clicking or tapping depending on the device being used.

Be aware that some hover-related user hints are simply not available to touch interface users.

Device support

Smaller devices such as most phones are too small to be adequately usable. For example, your Apple Watch is sadly unlikely to provide a very good django CMS editing experience.

Older devices will often lack the performance to support a usefully responsive frontend editing/administration interface.

The following devices are known to work well, so newer devices and more powerful models should also be suitable:

- iOS: Apple iPad Air 1, Mini 4
- Android: Sony Xperia Z2 Tablet, Samsung Galaxy Tab 4
- Windows 10: Microsoft Surface

We welcome feedback about specific devices.

Your site's frontend

django CMS's toolbar and frontend editing architecture rely on good practices in your own frontend code. To work well with django CMS's responsive management framework, your own site should be friendly towards multiple devices.

Whether you use your own frontend code or a framework such as Bootstrap 3 or Foundation, be aware that problems in your CSS or markup can affect django CMS editing modes, and this will become especially apparent to users of mobile/hand-held devices.

Known issues

General issues

- Editing links that lack sufficient padding is currently difficult or impossible using touch-screens.
- Similarly, other areas of a page where the visible content is composed entirely of links with minimal padding around them can be difficult or impossible to open for editing by tapping. This can affect the navigation menu (double-clicking on the navigation menu opens the page list).
- Adding links is known to be problematic on some Android devices, because of the behaviour of the keyboard.
- On some devices, managing django CMS in the browser's *private* (also known as *incognito*) mode can have significant performance implications.

  This is because local storage is not available in this mode, and user state must be stored in a Django session, which is much less efficient.

  This is an unusual use case, and should not affect many users.

CKEditor issues

- Scrolling on narrow devices, especially when opening the keyboard inside the CKEditor, does not always work ideally - sometimes the keyboard can appear in the wrong place on-screen.
- Sometimes the CKEditor moves unexpectedly on-screen in use.
- Sometimes in Safari on iOS devices, a rendering bug will apparently truncate or reposition portions of the toolbar when the CKEditor is opened - even though sections may appear to missing or moved, they can still be activated by touching the part of the screen where they should have been found.

Django Admin issues

- In the page tree, the first touch on the page opens the keyboard which may be undesirable. This happens because Django automatically focuses the search form input.

1.4.3 How-to-guides

Note: This is a new section in the django CMS documentation, and a priority for the project. If you'd like to contribute to it, we'd love to hear from you - join us on our friendly Slack group.

Use our guides to learn how to do a specific task in the CMS.

Managing pages

Managing pages in Django CMS involves creating, organizing, and editing the structure and content of your website. Here's a step-by-step guide:

1. **Accessing the Page Admin Interface:** Select "Pages…" in the page menu of the toolbar.
2. **Creating a New Page:**
   - Click on "Add Page" to create a new page.
   - Enter the page title, select the page type (if applicable), and define the URL and other settings.
3. **Organizing Page Hierarchy:**
   - Arrange the page hierarchy by creating parent-child relationships between pages.
   - Use drag-and-drop functionality to organize pages in a tree-like structure.
4. **Managing Page Content:**
   - Click on the settings icon of a page to access its settings
   - Click on the eye icon of a page to access its content.
   - Use placeholders to add content. Edit or add plugins (text, images, forms, etc.) into these placeholders.
5. **Editing Page Settings:**

   Adjust page settings such as metadata (SEO-related information like meta titles, descriptions, keywords), publication dates, permissions, etc.
6. **Preview and Publish:**
   - Preview the page to see how it will look to visitors.
   - Once satisfied, publish the page to make it live on the website.
7. **Managing Existing Pages:**

   To edit an existing page, navigate to the Pages section, locate the page, and click on its eye icon to access its content.

Make necessary changes to the content, settings, or structure.

Tip: **Permissions:** Adjust user permissions in Django CMS to control who can create, edit, or publish pages.

URL Handling: Django CMS often handles URL routing automatically, but you can customize URLs and slugs within the page settings.

Versioning and Revisions: Django CMS keeps track of page revisions, allowing you to revert to previous versions if needed.

Managing pages in Django CMS involves a mix of structural organization, content creation, and configuration adjustments. The system is designed to provide a user-friendly interface for content editors to manage the website's structure and content efficiently.

Managing versions

Note: This section only applies to installations such as the django-cms-quickstart project which use the djangocms-versioning package for versioning. django CMS' modular architecture allows for other versioning packages, too.

Django CMS keeps track of page revisions, allowing you to revert to previous versions if needed. Here's a step-by-step guide on how to manage versions:

1. **Accessing the "manage versions" view:**

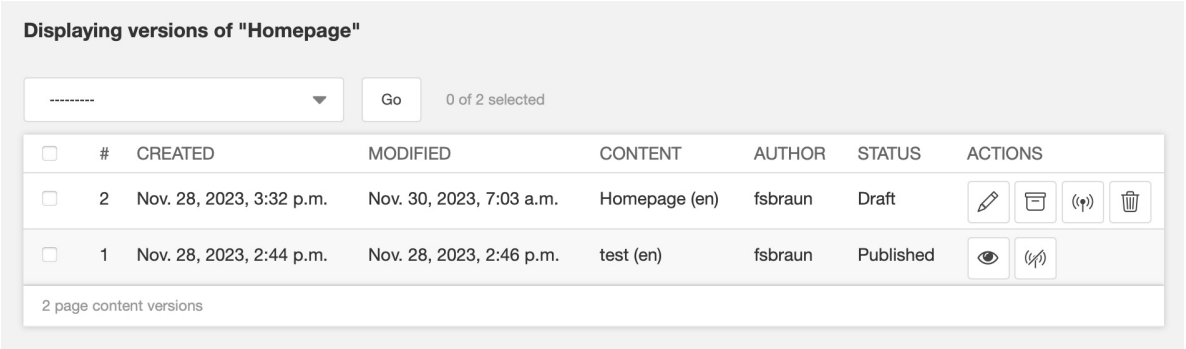
   Either

   - Select "Pages…" in the page menu of the toolbar and look for the page the versions of which you want to manage.
   - Click on the status indicator to open the dropdown menu
   - Select "Manage versions…"

   or

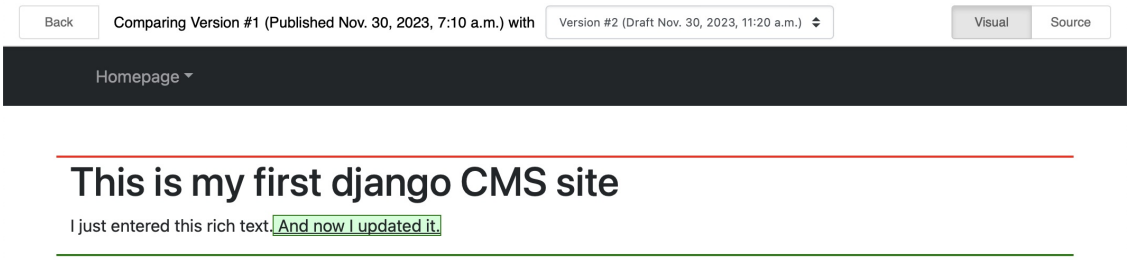
   - Preview or edit the page the versions of which you want to manage.
   - Click on the version menu and chose "Manage versions…"

2. **Inspecting all versions:**



   - Created and modified dates
   - Title and language
   - Username of the author
   - Status: Draft, Published, Unpublished, and Archived
   - Action buttons

3. **Acting upon versions:** Actions differ by status of the version. They include
    - Edit (pencil): Edit this version (draft) or create a new draft from this version (published)
    - Archive: Mark this draft as archived for later usage
    - Revert: Create new draft from this archived or unpublished version
    - Publish: Publish this draft
    - Unpublish: Make this published version unavailable
    - Delete: Delete this draft version
    - View: Preview this version
4. **Comparing two versions:**
    - Select exactly two versions to compare by checking the box on their left side
    - From the pull-down menu marked ------- select "Compare versions"
    - Click "Go"



Note: Only draft versions can be deleted. Outdated versions are kept on purpose.

Managing versions in Django CMS allows to understand the history of content. Drafts can be archived for later reuse. All versions currently public or once public are marked "published" or "unpublished", respectively. The content of versions can be compared with each other.

Managing media files

In Django CMS, managing media files such as images, videos, documents, etc., is often handled using a popular Django package called "django-filer." This package integrates seamlessly with Django CMS to handle file management efficiently. Here's a step-by-step guide on managing media files in Django CMS using django-filer:

1. **Accessing Filer in Django CMS Admin:** Once installed, you can access the django-filer functionalities through the Django CMS admin interface.
2. **Uploading Files:** Content editors can navigate to the Filer section within the CMS admin to upload new files (images, documents, videos, etc.). They can create folders, upload files, and organize them within the file manager provided by django-filer.
3. **Inserting Files into Content:** When editing content in Django CMS, content editors can easily insert files (images, documents) stored in django-filer using plugins or specific placeholders designed to handle media elements. For instance, they might use a "File" or "Image" plugin that allows them to select files from the django-filer library and place them within the content area.

4. **File Management:** Django-filer provides features for renaming, deleting, moving, and organizing files and folders within the file manager interface, ensuring a well-structured and manageable collection of media files.

By using django-filer, content editors can easily manage media files directly within the CMS interface, simplifying the process of adding and organizing various types of content elements throughout the website.

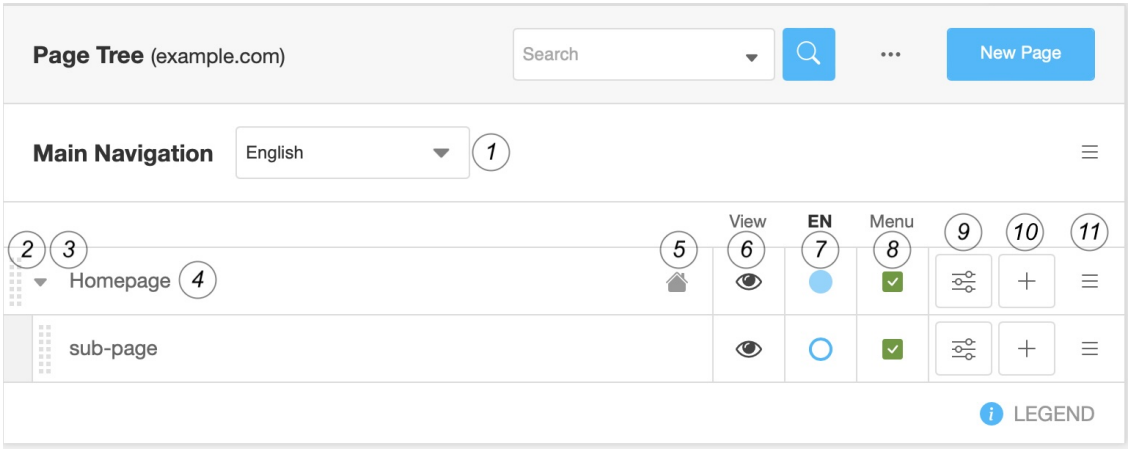
Managing redirects

By implementing redirects, you can retain the SEO rankings and authority of the old URL, transferring it to the new one. This helps to ensure that your website maintains its search engine visibility.

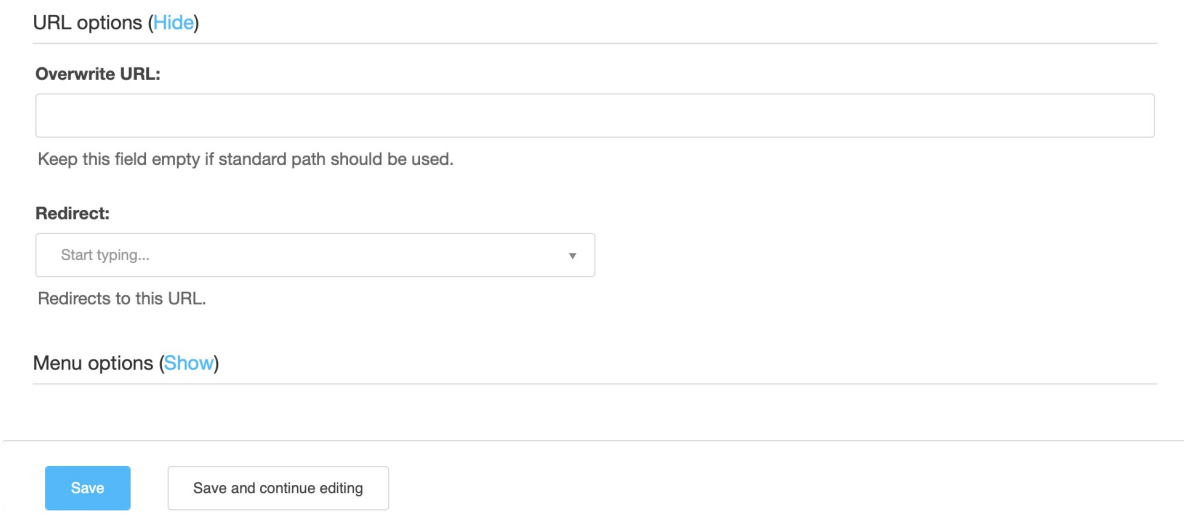
Django CMS supports redirects as part of the *Page settings*. It happens on a per-language level. This means you can select different redirect targets for the different language contents of a page.

The simplest way to set redirects is:

1. **Accessing the Page Admin Interface:** Select "Pages…" in the page menu of the toolbar.
2. **Opening the page settings:**
   - Find the page you want to redirect in the page tree.



   - If the page is published, select "Create new draft" from the dropdown menu of the page status indicator (number 7).
   - Click on the page settings icon (number 9: three horizontal sliders),
3. **Changing the redirect setting:**



- Find the "URL options" section.
- Click "Show" to open the "URL options" section
- Change the "Redirect" setting by selecting the page which the current one should be redirected to
- Close the setting by clicking "Save" at the bottom

4. **Publishing the change:**
   - Select "Publish" from the dropdown menu of the page status indicator (number 7)

Note: Changes can only be made to draft page contents and will only take effect once published.

Managing redirects in Django CMS happens on page level. The system is designed to provide a user-friendly interface for content editors to manage redirects and thereby ensure keeping SEO rankings.

1.4.4 Reference for content editors

Note: This is a new section in the django CMS documentation, and a priority for the project. If you'd like to contribute to it, we'd love to hear from you - join us on our friendly Slack group.

Standard plugins

Each site has its own set of installed plugins. This reference is based on the plugins installed with django CMS quick-start.

django CMS groups its plugins into categories. We list the plugins by category.

Generic

Text
The text plugin is a simple yet versatile plugin used for adding and editing text content. It allows you to directly insert formatted text, such as paragraphs, headings, lists, and links.

Alias
The Alias plugin is a powerful tool that enables content editors to display certain content on many pages without duplicating it. The Alias plugin let predefined content blocks appear at its position by linking them. If the alias content is updated, the linked content also changes.

Frontend

Note: This section only applies to installations such as the django-cms-quickstart project which use the djangocms-frontend package.

The frontend plugins are part of the djangocms-frontend package, which might (or might not) be installed on your site. Its purpose is to provide web site components like sliders, accordions, etc. for content editors. Thóse components in most cases are implemented by the deisnger of the web site. The contend editors can use the component at any place on the site.

Accordion
Use the Accordion plugin to build vertically collapsing accordions.

Alert
Provide contextual feedback messages for typical user actions for a handful of contexts: success, warning, danger, info, ...

Badge
Provide small counters or pieces of information for a handful of contexts: success, warning, danger, info, ...

Blockquote
Quote blocks of content from another source. Optionally provide a source.

Card
Structure content with a flexible and extensible container with multiple variants and options

Card layout
Organize several cards into one layout

Carousel
Cycle through elements, images or slides of text, like a carousel.

Code
Present (computer) code blocks

Collapse
Toggle the visibility of content

Container
Contain, pad, and align your content within a given device or viewport for responsive designs

Editor note
Mark contents visible to editors only

Figure
Display related images and text

Heading
Add headline with optional anchor.

Icon
Give visual clues by adding icons

Jumbotron (deprecated)
Showcase your hero unit

Link / Button
Reference and link other contents

List group
Displaying a series of content flexibly. Modify and extend them to support just about any content within.

Media
Construct highly repetitive components like blog comments, tweets, and the like

Picture / Image
Show images from the media library

Row and Column
Build responsive layouts of all shapes and sizes thanks to a twelve column system, six default responsive tiers

Spacing
Add horizontal spaces around the child content

Table of contents
Create a table of contents for all sections starting with the "Heading" plugin.

Tabs
Creat a tabbed interface for content that is only shown when the corresponding tab is activated.