Django Church Documentation

Release current

Django Church

February 04, 2016

Contents

Contents:

CHAPTER 1

Introduction

1.1 Welcome to Django Church!

If you're reading this, then it's likely that you're either thinking about using Django Church for your small Church website project, or you've already made the leap! In either case, you're most welcome! This documentation represents the general user manual for managing and maintaining your site's content once it's been set up. If you're after technical documentation about setting up your own Django Church instance, you'll need to go take a look at our developer docs. On the other hand, if you want a whistle-stop tour of Django Church's features and interface, you've come to the right place!

This guide starts by exploring Django Church's User Interface (UI). It then takes a look at some basic processes like logging in and changing your password, and lastly covers detailed walk-throughs of each app in turn.

Before you get stuck in we just wanted to say that this guide is not complete! It's a perpetual work in progress. As people ask questions and discover features we haven't fully outlined, we'll add our responses. If you have a query, or have spotted a gap in the guide, please do let us know. The best way to get in touch is by emailing support@blanc.ltd.uk

Onward!

1.2 A bit of background

1.2.1 What's in a name?

Django Church is so called because it's a Content Management Systems (CMS) for small Church websites, built through a web-application framework called Django, which is an open source tool programmed through the open source programming language called Python.

1.2.2 Why do we need Django Church? Aren't there lots of other content CMS options out there?

Good question… we'll overlook that I've just asked it myself. There are many, many, many CMSs out there designed to deliver Churchy-type websites. And there's an order of magnitude more designed to deliver websites generally.

Django Church is non-denominational, and has been developed in association with and funded by the Church of England: Birmingham to enable the future development of small Church websites in a more cost-effective and collaberative way. But, because of its open source nature, anyone from any Church, anywhere in the world can download and use the Django Church code, free of charge.

What sets Django Church apart? There's a few USPs…

1.2.3 It's open source

We could go into a lot of detail about what makes something "open source" - we could type for hours in fact. We've written a paper about open source systems which covers off the core details, but in short, you can think about it like sharing a cooking recipes. Someone develops a recipe and shares it with a few friends. Those friends take the recipe, tweak it, they might improve it but when they are done they share it back amongst their friends, who in turn take, amend and re-share it. In an ideal world, open source philosophy represents a cycle of continuous free and open shared improvement.

Now, amongst the other CMS options there are others built on open source foundations, but few of them - if any - are open source in the purest sense of the word. The majority work on a Software as a Service (SaaS) basis; you rent the site from the provider. They've used open source code and built a proprietary, locked-in CMS. As soon as you top paying rent, your site goes away. We don't think that's the way the Kingdom should work.

Django Church is fully open source. You can visit our Github repository and download it, tweak it, re-share it... with the right skills, you can make it exactly what you need it to be. And if you don't have the skills we can deploy and host it for you. At any point in the futre, you can walk away with your site, and spin up your own copy with the hosting provider of your choice. It's that simple.

1.2.4 What's Python?

Python is the programing laguage used to build Django

Python is a serious programmer's language. It typically doesn't attract 'script kiddies' or designers who've picked up other languages like PHP, and learnt what they need to get by. It means that the average Python programmer is significantly more advanced in their understanding and experience of development, re-enforcing point 3 above.

Python enforces consistent programming style, meaning it requires you to write code in a human-readable form, which again reinforces the transferability of code between developers in point 3 above.

Python was a software language long before it was used for web development. It's a big language designed to solve big problems (see NASA, Google etc.), as a result its support is wide and reaches into industries the world over.

Python scales well. It's the case that with lesser programming languages like PHP, the bigger the project, the more cumbersome and unstable the platform becomes. Python on the other hand doesn't suffer from those kinds of bottle-necks.

1.2.5 What's Django?

Django is a 'rapid prototyping' framework, aided by a huge library of tools and modules pre-built and ready to go. If you want users of your site to be able to log in to the site, you don't need to build the user-model that manages that process. The models to manage that feature can be added to the project with a line of code. As a result, it's fast to develop within, enabling rapid turn-around on projects.

Django is incredibly stable and isn't prone to falling over under pressure.

Django doesn't make assumptions about your project. Look at Wordpress for a moment. It's designed to do a fixed set of tasks. It doesn't matter if you don't want to make use of those features, they're still there, clogging up your work flow. Equally, if you want it to do a task that doesn't feature in its list, you're stuck. Django lets us build exactly what a client needs and then expand on that long-term. It's flexible, lean and agile as a result.

It's being used now, it's not a bleeding edge piece of technology that's only supported by three guys in a garage somewhere in Kent. It's used by mass-market websites like The Guardian, The Onion, Instagram, The Washington Post, NASA and PBS to name just a few.

Django is fast and scalable. There are examples of Django site's that draw millions of hits a day without blinking.

CHAPTER 2

Getting started

So here's the basics. The next few sections should equip you with all the core information required to learn everything else. We cover everything else *too*, but if your a quick study, the next few sections will see you right.

NOTE: Everything that follows assumes that you've first started by opening up the web-browser of your choice. Everything after the section called "Logging in" assumes that you have, in fact, logged in.

2.1 Logging in

1. Start by visiting the home page of your website. Let's, for the sake of argument, say that your site's URL is http://www.example.org
2. To get to the administration dashboard (admin for short) all you have to do is add /admin/ to that URL - http://www.example.org/admin/ - and hit return.
3. You'll then see a login screen with "Username" and "Password" fields. Type these details in and click "log in."
4. If all is well, you should now be looking at a page headed "Django Church administration"
5. If this is the first time you've logged in, it might be a good idea to update your password with something super-memorable and equally secure. You can do this by clicking "Change password", located in the top right of the screen. See the "Changing your password" section below for more.

2.2 Apps and models

All the functionality in Django Church breaks down into individual applications (apps) which deal with specific areas of functionality. The apps installed with the core Django Church package which you have access to, are listed in white text in blue bars. These are things like "Assets," "Events" and "Pages".

Beneath each app, you'll see individual models which relate to more granular areas of functionality. typically these contain lists of things, so for example, the "Images" model under the "Assets" app contains a list of all the images that have been uploaded for use on the website.

You can click on the App name which will drill you into a refined view of the models available to you. You can also click directly the model names to jump to their list views.

Django Church allows admin Super-users to restrict access of particular features of the site to other users of the site. The typical example might be that a several users are commissioned to keep the website up to date - and so need access to edit pages etc. - but won't need to add events to the system.

Knowing this is important, in that the apps and model you see when you log in, will depend on the actions your profile has been given access to.

2.3 UI basics

Django Church's UI is entirly consistent. If you see a button called the same thing as one you've seen elsewhere, it's a sure bet that'll do what you expect it to do. In our experience, this means that once a user has understood one section of the site, the rest of the system tends to be an open book to them.

It's worth noting that the majority of the UI conventions we'll talk about here are stock Django conventions which we inherit. Here's a few of the UI conventions it's helpful to bare in mind...

2.3.1 Required/Options

One of the simplest things to point to, one of the easiest things to overlook. when you're looking at a form, let's use the news post form (Home > News > Posts > Add post) for example. Look at the the field labels running down the left side of the form. Note that "Title," "Category" and "Date" are all **bold** while "Image" and "Teaser" are not. That's because **bold** field labels denote required form elements while non-bold ones are optional.

2.3.2 The green +

In short, a green + symbol next to an object gives you quick access to dialogues that you add one more of this kind of thing to the system. You can find them on each model row on the admin dashboard, giving you quick access to adding things without having to first click through to the list view.

You may also see them within larger forms where apps and models are cross-referencing each other. A good example of this is to be found with the "Add image" form in the Assets app (Home > Assets > Images > Add image). Alongside the "Category" combo-box, you'll see the green +, clicking it will let you quickly add a new category to the combo-box without having to back out of adding the image and go back into the "Categories" model.

2.3.3 The three saves

At the bottom of every form you'll find the same three save options, and they all have their uses...

1. "Save," typically high-lit in blue. Clicking this button saves the form and takes you back to the model list.
2. "Save and continue editing" is incredibly useful. It's like hitting control-S while you're working on a Word doc; it saves the contents of the form, but keeps you on the form so you can continue your work without fear of loosing a large volume of work.
3. "Save and add another" saves the form and then takes you to a fresh, empty form ready for you to add another. This is particularly useful when you are adding multiple instances of a type of thing in a single session.

2.3.4 What? No cancel button?

Yep; no. Django doesn't bother cluttering up your window with redundant buttons. If you've started something and you realise you don't want to bother saving it, just navigate away from the page by hitting your browser back button, of clicking the pebble-trail near the top-left of the screen. Nothing is saved until you save.

2.4 Changing your password

It's a really, really good idea to keep your password safe, secret and as cryptic and/or random as you can manage. We can't help you with that bit, but here's how you get there...

1. From any page within the admin, click "Change password" in the top-right of the screen.
2. You'll be presented with a form, first of all type your "Old password," this will confirm that you are who you say you are.
3. Then type your "New password."
4. Then type it a second time to confirm that you definitely got it right.
5. Click "Change my password" and you're done!

2.5 Where should I put my content?

This question isn't strictly about the use of the CMS, but about content and where you should keep it. It might seam obvious but it's often the case that, even though the apps are all labeled as to what their function is, user put content into the wrong area of the system. Sometimes event content ends up in a news story because the user rationalises that they want to tell a story about it; or more commonly, bite-sized news content ends up in general page content, over-complicating navigation and cluttering up the place.

The reality is, all three main content apps - pages, news and events - are capable of containing images, descriptive text, location information and date/time information. As a result, it's not always clear where a given bit of content should live.

We do a lot of Information Architecture (IA) work, and this issue goes to the heart of that discipline. It's complex and it's another subject I could write for hours on, but for a novice, the basic thrust can be summed up in one question: **What is the most important thing I need people to understand about this bit of content?**

If the answer to that question is "the most important thing about this bit of content is this future date," it's probably an event, and so the events section is the right place for that content.

If the answer is "it's about telling people about a thing that happened or is going to happen" -but the date is secondary, it's probably news.

And lastly, if the answer is "it's about making sure people find this content because it's about who we are," then that sounds like a page, at least because pages automatically appear in the site's navigation, so it elevates it in terms of perceived importance.

There are no absolute rules in all of this, and users typically find their way around these themes over time. The important thing is consistency.

2.6 URL character limitations

A note on best practice for constructing URLs. A number of the sections below deal with URLs for object, both uploaded to the system, and defined by it through "slug" and "URL" fields on particular forms.

If you remember one thing, remember this: **don't use spaces in slugs and URLs. Use dashes - or underscores _ instead.** It's likely that constructing a slug or URL with spaces will break things for at least some users on certain browsers.

So what characters are safe? In short, all alpha-numeric characters, plus a few punctuation marks. In practice you should limit your characters to…

abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ 1234567890-_./

…and where possible, don't use upper-case letters at all.

2.7 WYSIWYG text editing

What You See Is What You Get, or WYSIWYG (pronounced "wizzy-wig"), is the label we apply to tools which write HTML code for you online. They give you a Word-like experience of editing content, without having to bother with learning to format text through abstract HTML tags. There are many WYSIWYG editors out there, but Django Church uses Redactor. It strikes a great balance between robustness and offering common tools which most users might need.

There are three places where you can find WYSIWYG text editors within Django; events, news and pages. The rules are the same no matter where you are editing content, so rather than duplicate this section of the guide three times, this is your single reference point.

2.7.1 Limitations

The first thing to say is that WYSIWYG text editing is not an exact science. No one editor has it exactly right, and so as developers you have to choose the editor that best fits your project. However, this does mean that occasionally, you might need to look under the hood of the editor at the HTML it's writing for you. If that prospect scares you, don't worry, that's what support is for!

2.7.2 Best practice in writing your content

In an ideal world, users of the system would type their content directly into the context where it was being used. In practice people tend to write their content in Word, and then copy/paste it into the web browser. The problem here is that Word uses tags that look like HTML to define its text formatting. Unfortunately while the tags look like HTML, they aren't and pasting direct from Word to a WYSIWYG editor can scramble your text completely. If you do write your content in Word, it's worth copying and pasting it into a TXT editor like Notepad, then copying it again from Notepad and only then pasting it into your WYSIWYG editor. It sounds a bit contrived, but it will have the effect of stripping out all the HTML-like tags Word uses.

If you have to use an external editor, consider using an app like Notepad by default. The reality is, all your formatting of headlines and bullet-lists etc. will need to be re-done in the CMS anyway, so there's no benefit to styling your text before it gets to that point.

2.7.3 Formatting options.

1. Click Home > News > Posts > Add post
2. Half way down the page you'll see the **Content** section. Along the top of the large text field that follows, you'll say a row of button icons. These are the control dialogues for your content. From left to right...
   (a) **</>** is the HTML/Preview toggle. It lets your switch between the default WYSIWYG preview and the HTML beneath it. *Unless you know what you're doing, it's inadvisable to edit the HTML directly.* That said, it can be helpful to take a peak every now and then as it's really not that hard to pick up.
   (b) ¶ is called a "pilcrow" and is short-hand for "paragraph". This button gives you access to the various pre-defined headline and paragraph styles. Clicking it presents a drop-down list of the options rendered in an appropriate style - if not the fonts or colours they will finally be rendered in on your site. To apply a style, simply place your curser within the paragraph you want to affect, and select an option from that drop-down. Note that the style will effect the entire paragraph.
   (c) **B** makes selected text bold.
   (d) **I** makes selected text italic.
   (e) Struck-through **T** makes selected text struck-through

(f) The next two icons relate to bulleted and numbered lists respectively. Clicking either will create a list when on the line where the curser is sat, or alternatively, selecting a body of text will convert each paragraph selected into a bullet when clicked.

(g) The next two icons relate to indentation of paragraphs. to indent or outdent a para, simply place the cursor within the paragraph and click the appropriate icon.

(h) The **Insert image** (picture frame) icon lets you embed images inline with your text.

i. When clicked, it presents an input field which is expecting the full URL of the image.

ii. If you are linking to an image in the Assets app, all you need to do is to open a new browser window (or tab) and navigate to the assets list (Home > Assets > Images), and then copy/paste the URL of the desired image from the "Location" column.

iii. Clicking "Insert" embeds the image at the location of your cursor.

iv. *NOTE that the image should be as close to the dimensions as you intend to use them as possible.* Uploading larger image will result in very slow page-loads.

(i) The **Insert video** (play symbol in a rectangle) icon lets you embed videos from third party streaming services such as YouTube within your content.

i. When clicked, it presents an input field which is expecting the EMBED code which most providers supply.

ii. Clicking "Insert" embeds the image at the location of your cursor.

iii. For more information around embedding such content, please consult with your provider of choice for the best results.

(j) The **Table grid** icon enables you to create more complex tables of information within your page content.

(k) The **Chain-link** icon enables you to assign hyperlinks to text within you content.

i. First select the text you want to be clickable.

ii. Then click the icon, it will present input fields which define the URL you want users to see when they click.

iii. If you want the link to open in a new window, check the "Open link in new tab" checkbox

iv. Click "Insert"

(l) **Underlined A** allows you to pick a colour for the selected body text. *NOTE: Colours are assigned by the CMS Style-sheets. You should not need to assign colours manually in most cases.*

(m) **A in a Box** allows you to pick a colour for the background of the selected body text. *NOTE: Colours are assigned by the CMS Style-sheets. You should not need to assign colours manually in most cases.*

(n) The last but one icon lets you set the text **Alignment** for selected text.

(o) The **—** icon allows you to insert scores into text, which can be used to structure information on long pages of content.

Those are the building blocks. There's nothing like experimentation and exploration to help you get to grips with these tools, so we recommend sitting down with some sample content and playing with it.

CHAPTER 3

Applications

Here's the knitty-gritty. Here you will find detailed breakdowns of each app, and the fields of information you'll find when exploring them.

3.1 Assets

If you want to add an image to a page, or link to a PDF - a newsletter for example - from a page, you first need to upload it to the CMS. Assets is where you do that.

3.1.1 Asset models

Assets break down into two groups, files and images. Images are exactly as you'd expect - typically photos of one form or another with file-types such as JPG, GIF and PNG. Files are literally everything else; DOC, PDF, PPT, XLS, TXT etc. The models here are paired, in that files and images each have a counterpart category model where you can manage the categories for each. These two pairs are functionally identical as far as their management is concerned, so we'll use files as the example case here...

3.1.2 File categories

1. Click Home > Assets > File categories
2. Here are listed all the categories that can be applied to files.
3. To edit an existing category, click the category name.
4. To add a new category, click "Add file category".
5. The only field available for editing is the category title its self.

3.1.3 Files

1. Click Home > Assets > Files
2. Here are listed all the files currently available in the system.
3. **Each file presents the following bits of meta, from left to right...**
    (a) File title (click here to view)
    (b) Category (as defined by the File categories model above)

(c) The URL of the asset (clicking lets you preview the file)

4. To edit an existing file, click the file name.
5. To add a new file, click "Add file".

Add a file

1. Click Home > Assets > Files > Add file
2. **Detail**
   (a) **Category** defines the category the file as tagged with
   (b) **Title** defines the title of the image - for admin use
3. **File**
   (a) **File** defines the file to be uploaded.
   (b) Click "Choose File" and browse to the asset you wish to upload.
4. When saving and uploading an asset, the page will appear to be inactive for the duration of the upload. Please wait...

Editing files

1. Click Home > Assets > Files > [file title]
2. When editing an existing asset, this field contains two further elements. **Currently** denotes the file currently uploaded, clicking previews the asset; and **Change** which lets you replace the existing asset with another.

3.2 Auth

Short for "Authentication", Auth is where you manage user profiles, who can log in and what they can edit when they do.

3.2.1 Auth models

Auth contains two models, Groups and Users. The latter contains individual profiles for users with all their associated privileges, the former allows admins to assign privileges to defined groups. Users can then be added to those groups and inherit their privileges from them. Groups makes it very quick and easy to set up quite specific levels of privileges initially, and then apply those privileges in bulk quickly and easily.

3.2.2 Groups

1. Click Home > Auth > Groups
2. Here are listed all the groups that can be applied to user profiles.
3. To edit an existing group, click the group name.
4. To add a new group, click "Add group".
5. **Name** defines the tag that can later be applied to user profiles

6. **Permissions** defines what permissions a group has. The left column contains all the available permissions in the system, while the right contains all the permissions assigned to the current group.
7. Individual permissions can be moved between the columns by selecting items and then clicking the arrow icons; or by double clicking an item.

3.2.3 Users

1. Click Home > Auth > Users
2. Here are listed all the user profiles registered with the system.
3. **Each user presents the following bits of meta, from left to right…**
    (a) Username (click here to view)
    (b) Email address for this user
    (c) First Name
    (d) Last name
    (e) Staff status (a tick here indicates that the user has access to the admin dashboard)
4. To edit an existing user, click the username.
5. To add a new user, click "Add user".

Adding a user

Adding users is a two stage process. It's recommended that you undertake both in order to fully create a user profile in a single sitting, but this isn't strictly required…

1. Click Home > Auth > Users > Add user
2. **Username** is the name that a user uses to access the admin, must be 30 characters or fewer, consisting of letters, digits and @/./+/-/_ only.
3. **Password** and **Password confirmation** should be identical for verification purposes.
4. Click "Save and continue editing", the page will refresh with the second tier of options.
5. You will note that the username value previously filled in is reproduced here. The password will look broken - don't panic! It's not broken, it's encrypted so that other users can't abuse other user's passwords.
6. **Personal info**
    (a) **First name**
    (b) **Last name**
    (c) **Email address** for ease access and support.
7. **Permissions**
    (a) **Active** checkbox designates whether this user should be treated as active. Unselect this instead of deleting accounts.
    (b) **Staff status** checkbox designates whether the user can log into the admin site.
    (c) **Superuser status** checkbox gives you super powers! No, seriously it designates that this user has all permissions without explicitly assigning them.
    (d) **Groups** defines whether the user belongs to a group, inheriting that group's permissions.

(e) **User permissions** defines what permissions a user has in addition to inherited group permissions. The left column contains all the available permissions in the system, while the right contains all the permissions assigned to the current user.

(f) Individual permissions can be moved between the columns by selecting items and then clicking the arrow icons; or by double clicking an item.

8. **Important dates**

   (a) **Last login** denotes the date and time of this users last use of the CMS.

   (b) **Date joined** denotes the creation date of the users profile.

   (c) Both these fields can be edited, but are included largely for reference purposes.

Editing a user

1. Click Home > Auth > Users > [username]
2. has the effect of dropping you directly into the second of the two stages, point 5. under "Adding a user" above.

3.3 Events

Just FYI, the events section of the CMS is one of its more complex. structuring events generally isn't easy, and so we've sought to break it down into a structure that would make sense to most users - on both the admin *and* on the public sides of your site.

3.3.1 Events models

As a result we've split events as a general content type, into two groupes. Firstly, recurring events which happen every week - Sunday services for example - and secondly, special events which only happen once and/or less regularly than once a week.

3.3.2 Recurring events

1. Click Home > Events > Recurring events
2. Here are listed all the regular events currently available in the system.
3. **Each event presents the following bits of meta, from left to right...**

   (a) Title (click here to view)

   (b) Day of the week (Monday, Tuesday. etc.)

   (c) Published (indicates whether or not the event is visible to the public)

4. To edit an existing event, click the event title.
5. To add a new event, click "Add recurring event".

3.3.3 Add a recurring event

1. Click Home > Events > Recurring events > Add recurring event
2. **Title** defines the title of the event. Note that this is the publicly visible title users of the site will see
3. **Image** is an optional field that lets you define a key image to associate with the event. This is achieved by selecting an image from the Image Assets app through the combo-box.
4. **Description** defines the descriptive text about the event which is seen when users navigate through to the event's detail page.
5. **Day of the week** The combo-box defines the kind of breakfast you'd like on that day... kidding. It defines the day of the week the event happens on.
6. **Frequency** is a fuzzy field designed to give users an indication of how regularly the event happens. "Weekly" or "fortnightly" for example.
7. **Published** check-box defines whether the event is publicly visible on the website. Uncheck this instead of deleting it if the event is no longer needed.

3.3.4 Special events

1. Click Home > Events > Special events
2. Here are listed all the special events currently available in the system.
3. **Each event presents the following bits of meta, from left to right...**
    (a) Title (click here to view)
    (b) Start (the start date and time of the event)
    (c) Published (indicates whether or not the event is visible to the public)
4. To edit an existing event, click the event title.
5. To add a new event, click "Add special event".

3.3.5 Adding a special event

1. Click Home > Events > Special events > Add special event
2. **Title** defines the title of the event. Note that this is the publicly visible title users of the site will see
3. **Slug which refers to the human-readable portion of the event's URL once it's live.**
    (a) For example an event with the slug "coffee-morning", the resulting URL would be this "http://www.example.org/events/view/special-event/".
    (b) Note that standard URL character limitations apply here.
4. **Image** is an optional field that lets you define a key image to associate with the event. This is achieved by selecting an image from the Image Assets app through the combo-box.
5. **Description** defines the descriptive text about the event which is seen when users navigate through to the event's detail page.
6. **Start** and **End** which define the Start and end date and time.
7. **Published** check-box defines whether the event is publicly visible on the website. Uncheck this instead of deleting it if the event is no longer needed. Note that past events are hidden from the public site and so don't need to be manually unpublished.

3.4 News

Call it news, call it a blog, call it a journal for your church. What ever you call it, News is an area to store individual articles as a generic, date ordered, generic content type.

3.4.1 News models

News contains two models, Categories and Posts. The latter contains individual news articles, the former contains categories which can be applied to individual posts. These categories are automatically reflected in your site's navigation to help users find the kind of content they are most interested in.

3.4.2 Categories

1. Click Home > News > Categories
2. Here are listed all the categories that can be applied to news.
3. To edit an existing category, click the category name.

Add a category

1. To add a new category, click "Add category".
2. There are two fields available for editing…
3. **Title** which is the label visible to the public, so this needs to be short and descriptive of the kind of news users will find tagged with it.
4. **Slug which refers to the human-readable portion of the articles URL once it's live.**
    (a) For example as article with the slug "newsletter", the resulting URL would be this "http://www.example.org/news/2014/05/18/newsletter/".
    (b) Note that standard URL character limitations apply here.

Posts

1. Click Home > News > Posts
2. Here are listed all the news articles posted to the system.
3. **Each post presents the following bits of meta, from left to right…**
    (a) Title (click here to view)
    (b) Date (the post date)
    (c) Category
    (d) Published (denoting whether the post is visible to the public)
4. To edit an existing post, click the title.
5. To add a new post, click "Add post".

Add post

1. Click Home > News > Posts > Add post
2. **Title** defines the title of the post - publicly visible
3. **Category** defines the category the post is tagged with
4. **Date** defines the date at which the post is visibly published to the site. Note that this value can be in the future as well as the past, meaning it can be used to set a post to 'go live' automatically at a particular date and time.
5. **Image** defines an optional key image for the post which is automatically included on the news pages where the story appears.
6. **Teaser** defines an optional abstract for the story. This is used in things like feeds and news lists. As the name suggests it should be a single sentence which describes the post. It can not contain HTML tags for formatting.
7. **Content** represents the body of the article. Please refer to the section called "WYSIWYG text editing" above for more details on how to get the most out of this field.
8. **Advanced options**
    (a) **Slug** which refers to the human-readable portion of the articles URL once it's live. For example as article with the slug "newsletter", the resulting URL would be this "http://www.example.org/news/2014/05/18/newsletter/". Note that standard URL character limitations apply here.
    (b) **Published** checkbox defines whether the post is visible to the public or not. This option can be used to keep private posts which are in draft-form and not quite ready to be seen by the world at large. It can also be used to retrospectively hide posts.

3.5 Pages

In the majority of cases, pages within a site will account for the vast majority of your content. The Pages app gives you the tool, not only to manage the content on a given page, but to manage the structure of those pages and how they appear to the user.

3.5.1 Managing page and page structures

1. Click Home > Pages > Pages
2. Toggle between "Tree" and the traditional "Grid" view of pages by clicking the "- view" button in the top-right of the screen.
3. The Pages app has only one model, its self called 'Pages'. It differs from many of the other apps, in that the default "Tree view" of the app's contents is functional. The default Django Church build comes with a rough outline of some of the pages any given Church might find valuable. This also has the benefit of giving you a loose structure to play with while you're getting used to the way Pages work.
4. **Each row in the "Tree view" represents a page on your site. Each row (for example /sunday/ – This Sunday (edit)) will co**
    (a) **/sunday/** which relates to the pages slug within the site's URL
    (b) **Sunday** which relates to the pages title as it will appear in navigation
    (c) **(edit)** click to edit the page's content

(d) If a page has related sub pages, these appears as 'nested' trees within their parent. Sub-pages are indicated by the presence of an arrow to the left of the page. Clicking this arrow shows/hides the sub-pages which, when visible, are indented to indicate their parentage.

3.5.2 Ordering pages in "Tree view"

The "Tree view" its self lets you organise the pages within your site through a simple drag-and-drop mechanism.

1. Click Home > Pages > Pages
2. Find the page you want to organise
3. Click and hold the page's URL or Title elements
4. Dragging the page up and down the list will result in horizontal, blue lines appearing indicating where in the list the page will end up should you let go at a given location.
5. Release the page to drop it into the desired location.

3.5.3 Adding a page

1. Click Home > Pages > Pages > Add page
2. **URL** defines the portion of the page's URL that appears after your site's domain name, **/about/** for example where the full URL is http://www.example.org/about/. Note that the URL requires leading and trailing slashes: /
3. **Title** defines the publicly visible title which users will see on the site. Try to keep this short and to the point.
4. **Navigation gives you mechanisms to manually control where the page appears in the site.**
   (a) **Parent** enables you to define the page's location directly, rather than using the drag-and-drop tool found in the "Tree view" noted above. Clicking the combo box with give you a list of ALL the pages in the CMS. Moving a page is as simple as clicking the page you want to be the current page's parent.
   (b) **Show in navigation** checkbox enables you to hide pages by unchecking. Note that the page is *still live*, but won't appear in public navigation. This means that users of your site can still see the page if they have it bookmarked or, for example, if you email them a link to it.
5. **Content** defines the body content for the page. See *WYSIWYG text editing* for more detail on text options here.
6. **Advanced option**
   (a) **Template name** defines the design template used to display the pages content on the public site. Unless you've had additional page templates designed and built by a development partner, "Default" will be the only option available to you.
   (b) **Published** check-box defines whether the page is publicly visible on the website. Uncheck this instead of deleting it if the page is no longer needed.

CHAPTER 4

Support

This documentation was produced by Paul Evans, creative director at Blanc Limited, the open source publishers of Django Church.

They are accurate at the time of publishing, but we're sure that you good people will find gaps and will have questions we havn't anticipated. With that in mind, please don't hesitate to direct any questions you have to support@blanc.ltd.uk and we'll see how we can assist you. More over, when we are asked a question about the platform that isn't covered here, we'll add the answer to these docuemnts with the aim of growing a developing its content over time.

CHAPTER 5

Content & Structure

The other sections of this guide deal, largely, with the matter of using the Django Church CMS. It's basically the user guide to all the knobs and levers behind the curtain that tell the site what to show and where. But, if the CMS is the puppeteer's frame and strings, the puppet is the content itself.

The following sections have been written with the aim of giving you some help and direction in thinking about the structure of your site and how you should approach your content. There are no *absolutely right* ways to go about this process. There are, however, some *reliably wrong* way to approach it, which we'll help you side-step.

5.1 Your target audience

5.1.1 Who is your site for? Internal or external?

When thinking about your site, its structure and its content, this is arguably the most important question to ask yourself. There aren't any right or wrong answers to this question - not inherently anyway. But if you set up your site based on internal structures, roles or needs, you can't then be surprised when people on the outside looking in aren't engaged by what they find. So with that in mind, here's another way of asking tat question...

Is your site primarily about communicating to your existing Church members, or is it about reaching out to people outside your Church with whom you don't currently have a relationship?

5.1.2 The shop front

Often, the answer is "both." So how do we structure a site in a way to cater for both? It maybe helpful to apply the analogy of a shop front. The window of a high-street shop is largely designed to get passers-by to walk through their door. It's about presenting potential patrons with an enticing view of what they might find within. It's about showing them a little of everything such that no matter who they are, they'll see something that might be relevant to them.

Once they are through the door though, the presentation shifts. The ownness then falls on the patron to find what they are looking for using the way-finding devices the shop owner makes available to them. At that stage though, the patron has bought-into the shop, they are through the door, and the way the shop owner communicates with them changes.

The same applies with the people your church has relationship with. If your website as a whole is the shop, then its homepage is its shop-front. Your Church members are the patrons who have already bought into you and so you can afford to communicate with them differently. Viewing the homepage of your site, it's quite easy weight it in favour of the person outside looking in, while structuring the deeper content in favour of those already inside looking out.

5.1.3 80/20

Consider a 80/20 split. On the homepage, 80% of your content could be focussed on the passer-by, presenting them with snippets of who you are, where and when you meet, and possibly ways into both your Church body, and the website. The other 20% then gives your congregation easy access to the content they are looking for, the nuts and bolts of church life which passers-by would not be so concerned with.

But once you get past the homepage, the split flips-over; 20% of the depth of your content deals directly with the core of the gospel and the needs of people outside your Church, while 80% deals with notices, meetings, downloadable rota PDFs etc.

This way, you grab the attention of people who might be looking to your website for a flavour of who you are and/or interested in what you believe; while simultaneously catering for the day-to-day needs of your congregation who will be more inclined to click furth into the site to find what they are looking for.

5.1.4 Bottom-up or top-down?

Another theme to consider is who the subject of the content is. Is your site's structure built around the user and the needs that they might come to the site with (bottom-up) or is it about your Church and what you want to communicate to the user (top-down)?

User needs

A bottom-up approach puts the user in the driving seat of their experience. It puts areas of content about their experience front and center, while content that's overtly about your organisation's agendain the periphery. At the front end of the site might ask the user questions like *how can we help you?* Or *what would you like to know?*

It's all about allowing them to find the information they are after organically, and then using secondary mechanisms to subtly communicate the message you'd like them to leave with.

And that's the risk with this pathway; a bottom-up approach always runs the risk that the user walks away with the information they need, *but not the message you wanted to leave them with.*

Organisation message

A top-down approach guides users through the message you want to deliver, and then relates the user's needs to that message. So it's "we have a heart to serve the community, which is why we run a free Mothers and Toddlers drop-in" as opposed to the bottom-up "Come to our free Mothers and Toddlers drop-in which we run because we want to share God's love with our community."

The risk here is that it can sometimes feel heavy-handed and inward-looking. It's possible that users miss the services you might be inviting them to, because it's addressed through a framework that address the organisation first and obscures the user's pathway into it.

Best practice

As with other areas here, the tipping point will be somewhere between the two and different for everyone. However the predominant trend today is to lean towards a "user needs" approach, pulling in elements of overt organisation messaging where a clear direction is required.

5.1.5 So, where does this lead us?

The key thing is to have a clear view on who your website is targeted at, what information you have that they might need, and some thoughts on how you might balance those needs with the message you want to communicate. The next thing to do, is start to assemble the bones of your site.

5.2 Structuring your website

Okay, let's get some jargon out of the way. *Information architecture* is the art of organising and structuring information in a logical way which balances both the ease with which users can find the information they are looking for, and your desire to communicate a message to them. This works in two main contexts; firstly the naming and ordering of the pages which comprise your site with an architecture tree (just like a directory tree on your computer), and secondly how the content is ordered with each page. As was observed in the section entitled "Bottom-up or Top-down" in the previous section of this guide (If you haven't read "Your target audience" yet, it's definitely worth a look), these two needs often work at odds to one-another, so it's definitely worth making sure you are clear as to the approach you want to take.

But don't worry. One of the great things about Django Church, is that very little is ever set in stone. The structure of your pages, what you call them and where they appear in the site's architecture tree, can all by controlled and changed as required. In practice what this means is that you will have the flexibility to try different structures and organisations over time. If a page isn't working in a particular location, no worries, with a few clicks, you can move it to where-ever it needs to appear. The same applies to you your content, Django Church gives you the ability to edit the content on each page of your website, enabling you to develop and refine it over time.

5.2.1 Tip: Buy some post-it notes

This is typically how we start to structure a site...

1. Buy post-it notes. Lots of them. Branded, unbranded, any colour, any shape. Knock yourself out!
2. Find a space with an empty wall and a table.
3. One sheet at a time, write an area of content you need to contain within your website. Common areas of content for church websites might be...
   (a) Vision
   (b) Mission
   (c) Services
   (d) Staff or Team
   (e) Contact details
   (f) *Keep going...*
4. As you write them, stick the post-its on the wall. It doesn't matter for now what order you put them in, or whether they are structured, just get them on the wall.
5. There are new daft ideas here. It's better to capture everything and represent it on the wall, than to dismiss it and forget about it. There'll be plenty of time to trash redundant content areas later.
6. At the point where you start running out of ideas, take a step back from the wall and try to get a feel for the bigger picture.
7. Where there's a common theme or an obvious relationship between post-its, move them and group them together.

8. You might find that as connections are made, more ideas spring up. Make sure you capture those too and put them on the wall.
9. As those groups of post-its grow, start to think about what you might call those groupings. Ask yourself two questions... 1. In one or two words, what would I call this so that members of the Church would know what it is? 2. In one or two words, What would I call this so that people outside the Church would know what it is?
10. If the answer to those question are the same, then great. If they are different, make sure you capture both. Put your answers on a post-it and stick it above the group that it referes to. It might be worth using a different colour post-it or pen so you can identify these grouping labels.
11. Aim to have as few groups as possible, but not so few that you can't give each group a name that clearly identifies what it contains.
12. Slightly more tricky, assign each group a number where number 1 is the most important area of the site to your target audience, number 2 is the second-most important, and so on.
13. Once you're happy, take a photograph of your wall to document it. This way, you can refer back to it without elements being lost. It also means you have the option to start from scratch, laying out the post-its in a different fashion to explore alternate structures.

And you're done! That's your architecture. The group names represent your primary, or 'top level' navigation, and all the notes beneath represent secondary pages within.

5.3 Some pitfalls

In our experience of working with organisations, all too often their content development strategy is inward looking. Now this isn't a problem, if the brief is for an internal resource - but in that case we're probably talking about an intranet, rather than a website.

5.3.1 Organisation structure as website structure

Typically they'll poll their internal departments asking what the site needs to be and/or do, in order to satisfy the needs of their external users. Now, there's nothing wrong with that question, you'll certainly get *some* helpful detail out of it. The problem most often encountered, is that if you take that feedback and try to represent it literally, you end up with a structure for your website that's based on the internal structure of your organisation - a structure that's likely to be entirely opaque to your average external user. It may be accurate, but how is a user to know that the information they are looking for is managed by Department A, or Working-group B or Committee C?

The example we often use to illustrate this problem is this: You're an employee of a large organisation consisting of many dosen departments. You've got some expenses you need to claim back and you've been told there's an expenses form you need to download from the website, fill in and return. Where do you look for it? Is it a Finance issue? Or Accounts? Or possibly Human Resources? Departments might have their own procedures for claiming expenses, so it could be your own department's responsibility. Any of those options might be a reasonable assumption, so where should that user start looking? Ultimately, it's not important to them who looks after the form, they just know that they **NEED** it.

5.3.2 A lack of focus

Secondarily, in polling departments, they'll often ask questions about what each department needs in order to be accurately represented through the website. The problem here is that if you ask any given department about how important *their department* is within their organisation, they probably won't say theirs is *the most* important, but they'll likely rank themselves in the top three. It's human nature to be protective of the things your passionate about, and as a result, most people will say "my department needs to be somewhere on the homepage, because users need to

know how to find us." Once more, if you were to try to represent all that literally, your website will end up being a confusing mash of mixed messages and conflicting calls to action.

In one extreme case of a national charity we worked with, these miss-steps resulted in possibly the most compromised, confused website structure we've yet seen. The home page included not one, but three duplicated instances of primary level navigation (by which we mean the first series of pages linking off the homepage, with the pages linking off *them* being referred to as secondaries, and so on…), each slightly different to the others. There were no less than four different search boxes, each delivering results for different departmental areas of content. And most confusingly, there was a raft of information designed to cater for internal users which duplicated their organisation intranet, which had no relevance to the majority of visitors to the website.

5.3.3 Choice paralysis

Another common issue is driven by the laudable desire to be really, really, really helpful. The administrator of the website wants to make sure that, no matter who you are or what you've come to the site to find, that goal is never more than a click or two away. It's a good aim, but one expression of that desire is to dump all of those options onto the homepage, allowing users to pick their way through and find the one option they need.

Interestingly, this approach actually works against the ultimate goal. It results in something we call *choice paralysis*, you can also call it "decision fatigue" or "overchoice", but the principle is the same. Studies have shown that, when asked to make a choice between a given number of like items, a threshold is reached where-by more choice ceases to be helpful and actually reduces the likelihood that an individual will choose *anything at all*.

In the context of structuring a website, presenting a user with too many options is likely to result in them making no choice at all, save one; to navigate away from the site and find what they are looking for else-where.

For many years it's been the case that developers of websites have sought to reduce the number of clicks needed to get to any one piece of content, a tendency driven largely by historically limited bandwidth. Pages took a long time to download, and so you'd try to present a wide profile of options at any one point to keep your site's maximum click-depth relatively shallow. Click-minimisation was therefor a functionally-driven necessity.

In the UK today, the average connection speed to the internet is now exponentially faster compared to 10 years ago, yet click-minimisation tendencies still define how we think about structuring sites. Click-minimisation is not a bad thing, we should always seek to eliminate redundant pages and clicks that don't deliver significant content, but there are other approaches which can now be used which limit the number of options presented to a user, guiding them through the site. These might result in more limited profile of options for a user to evaluate, leading to a potentially higher click-depth - but minimising choice paralysis in the process. If a user doesn't know what they are looking for, it's often better to present them with a series of ever-narrowing choices - each leading them closer and closer to their end-goal, rather than have all of those options available simultaneously at a higher level.

There's one caveat here, and that's mobile access. Bandwidth for mobile users is still relatively slow, and as a result click-minimisation strategies are still entirely relevant. If a sufficiently large number of your users are accessing your site from a phone or tablet, presenting a wide profile of navigation options might be the right solution. Not wishing to add more complexity, it's also worth factoring for screen-size. If a user is accessing your site through a device with a diminutive screen, your wider navigation structure might run off their screen, obscuring their view of the breadth of options available to them.

5.3.4 Ultimately…

This kind of confusion is a product of a mismatch of a site's target audience. It's the result of assuming the needs and priorities of the target audience, are the same as the needs and priorities of the people commissioning the website.

These are just a few of the things you might want to consider when structuring your website's content, but the solution is likely to be specific to you, and will be based as much on the way you feel about the matter as anything else.