Content modeling: Creating structure and navigation

contentful

Content modeling: Creating structure and navigation

Websites are still hugely important as part of the expanding range of digital experiences offered across many devices and platforms. There's no doubt their creation and maintenance take up a sizable chunk of a digital team's time, effort and brainpower. Unless you want a lot of unhappy editors, you can't just build the content model around the needs of developers. While it can be a juggling act in the creation stage, making the right decisions early on will keep both happy.

You also need to be future-minded and build in the ability to scale and change. We're big fans of Contentful's flexibility and illimitable infrastructure, but we understand that the initial planning and building phase can overwhelming and sometimes we need limits and structures to feel comfortable with our projects.

This white paper will explore three structures that can serve as a starting point to a navigation content model. Each is suited to different types of sites so it's as simple as choosing one and then making it your own. You can also come up with a hybrid – it's all about creating what best suits the project.

Before you even touch the content model, it helps to create or revisit your content strategy. The goal of a content strategy is to create a meaningful and consistent experience for the customer. The strategy will often encompass auditing, planning, structuring and testing content – at its core, a content strategy will be your essential structure.

Once you know your content strategy, you can start creating a content model. A content model consists of individual content types, and essentially gives structure and organization to the content by documenting all of the different content types involved in the project. It helps to think about the content types as various stencils, which will eventually come together as a drawing.

Understanding how to best model your content to fit a website is hugely important. There are numerous challenges that come with content modeling but a common one is being able to model website navigation that works well for both editors and developers. Content modeling navigation isn't easy. It can cause even the most proficient content modeling experts to become unstuck. But when it's done right, it becomes a power tool that improves workflow and productivity.

The goal of a content strategy is to create a meaningful and consistent experience for the customer.

THE DEVELOPER AND EDITOR PROBLEM

Problems arise in the fact that developers often want to return navigation in a single API call while editors want to see their information architecture reflected in their content hierarchies within Contentful. During the process of content modeling, thinking of website navigation as a visual representation of an information architecture is helpful to understand and map how users navigate around the website – and it keeps editors happy too. Navigation could simply be represented as a set of labels and hyperlinks; but that's just one technique of many.

More complex navigation systems may also require these label/hyperlink combinations to be a representation of hierarchy such as parent or child links. This will involve navigation that's not only vast in breadth, but also exist and extend several tiers deep, might exist across separate instances, and possibly repeated. For example, pages might exist but not be listed on the navigation, or a separate section of the website might need its own navigation.

It gets more complex when talking about other digital touchpoints because you're again adding in separate navigation menus. The mobile version might also have its own navigation, different languages might require a new navigation too. Some navigation could have sub menus, some might not.

With how complicated a navigation structure can get, it helps to first think about your content model in a generic way. When you have a few different methods to approach your content model, you can pick the one that will serve you best. In the next section, we'll explore some of these generic ways to approach content modeling that will keep both developers and editors happy.

UNDERSTANDING YOUR CONTENT

Having a full understanding of the content, usually through a comprehensive audit as part of developing a content strategy, is the best starting point for content modeling. This can look different for different people and feel free to get creative. To get a high-level vision, we've found that working first on paper, with multi-coloured post-its, or on a whiteboard can save time in the long run. You'll find during this process that the mental model might change as new information is gathered.

As Contentful is created for, and best used by, cross-functional teams, content modeling is also best done with cross-functional teams. You'll find that pulling together at least a team of designers, developers, and editors, while including senior decision makers, will go a long way in ensuring the final result will keep everyone happy.

It's okay to go deeper with the content model in this early stage. The mantra is that the better you know your content, the better your content model will be. The content model will ultimately represent the entire website's content, so making sure you include all aspects, including content that isn't directly visible to the user like SEO metadata, is important. Try to make decisions around flexibility and scalability early on; You'll save time later if you're already thinking about future iterations.

The content model should include content types that form topics; which could be events, articles, and products. From there, the content model should be broken down further to include well-defined topics. These topics are the substance of your content or what an app or website is about. For example, if you have a web page presenting articles, the articles would be the topics.

If you have a watch screen that shows the wearer their events, the events would be the topic. In addition to topics, you will create assemblies. Assemblies are essentially content types, but with less diverse fields. They don't describe what a site is about, but rather how it is put together. If you want to dive deeper into topics and assemblies, you can read more here.

TOPIC RELATIONSHIPS

This model is great if your content is purely topical content that is then parsed/rendered using a different set of things. Ultimately if the raw 'things' in your content are related to one another – and that is how you want this information to be represented – then this pattern may work for you. One example would be a hotel chain. Each hotel has a relationship with its rooms, which each have different services and amenities available. It doesn't matter where this content is being displayed (in-room or online), the relationships stay the same.

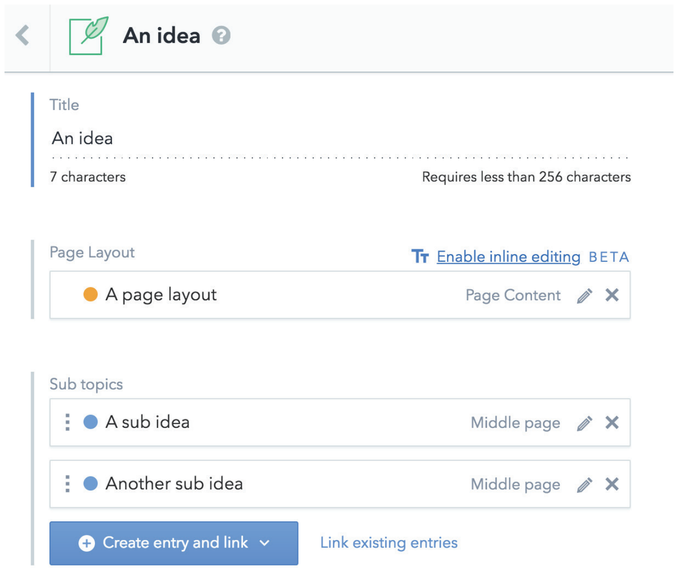
Here's some things to consider with this model:

- It's best when content is being used in different channels. For example, let's say you're creating a site for a hotel chain and you want to update the services page across all of the hotels. The services would be the topic that content creators need to create just once, before reusing it across all hotel sites. This removes the need to duplicate efforts.
- Navigation needs to make logical and conceptual sense. The relationship needs to reflect reality. Your model also needs to reflect the content strategy. The navigation needs to fit and grow with your content strategy, not the other way around.
- This generic model suits static sites, or sites that have some server side component.
- The topic relationships model is the sensible approach when thinking about how retail products relate to one another. You sometimes have related products or product variants that need representing.
- It's also the go-to for sites where you have products related to a product page. And your product page is related to another product page.

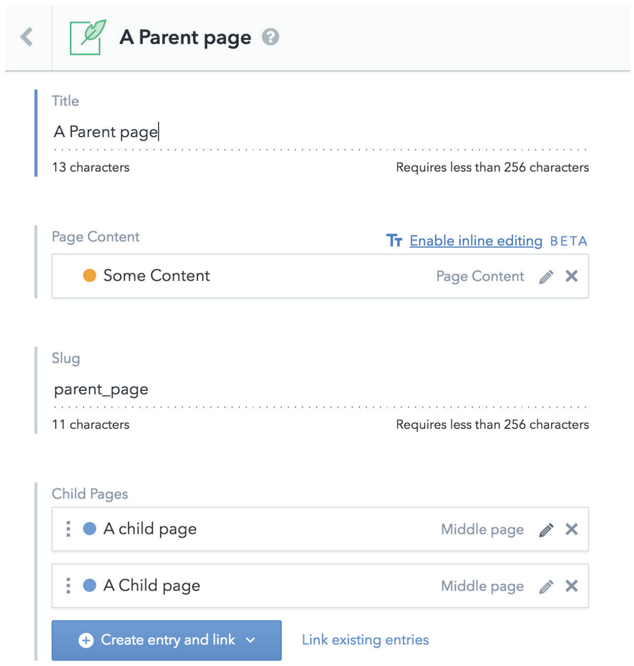
Exact implementations vary but the process is the same:

- Define the relationship types (parent, child, sibling)
- Define the validation rules around these relationships. You can have only one parent, siblings must be the same type etc.
- Add the relationship fields to topics
- Add the linked entries in the relationship fields

When generating navigation in this way, the relationship can be pulled at the same time as the main body of content. Reference fields with a linkType to an Entry will then create relationships, and the application can parse this easily.



An example of topics relationships in Contentful


A Parent page

Title
A Parent page
13 characters Requires less than 256 characters

Page Content

Some Content Page Content

Slug
parent_page
11 characters Requires less than 256 characters

Child Pages
A child page Middle page
A Child page Middle page

An example of assembly relationships in Contentful

ASSEMBLY RELATIONSHIPS

The assembly relationships model is the way content creators will put individual content types together to make web pages, documents, apps, or other content products. If the content is designed to be specifically displayed on a website, and the page assembly is the core way users access the content (via a URI for example), then assembly relationships may work best for you.

In this way, if your assemblies have children that are topics, they can also have children that define the relationships -- but it is important to note that the pages have relationships, not the topics themselves. That said, this could be combined with other relationships in the topics - for example, a looser relationship like 'related to'.

Here's what you need to consider with assembly relationships:

- An assemblies model is best when the content is channel-specific. Unlike topics, assemblies have site navigation data and fields for layout. They don't have any content. Boiled down to the simplest explanation, an assemblies model is best when pages are related to other pages.
- When you use an assemblies model, the implementation process is the same as topical relations, but will follow a stricter sitemap structure.
- A good example of an assemblies model site is when the page has a header navigation block, which might contain five pages. The navigation block is defining the relationship. The nesting of the page and navigation block mean they are one thing.

NAVIGATION CONTENT TYPE: USING PAGES OR TOPICS

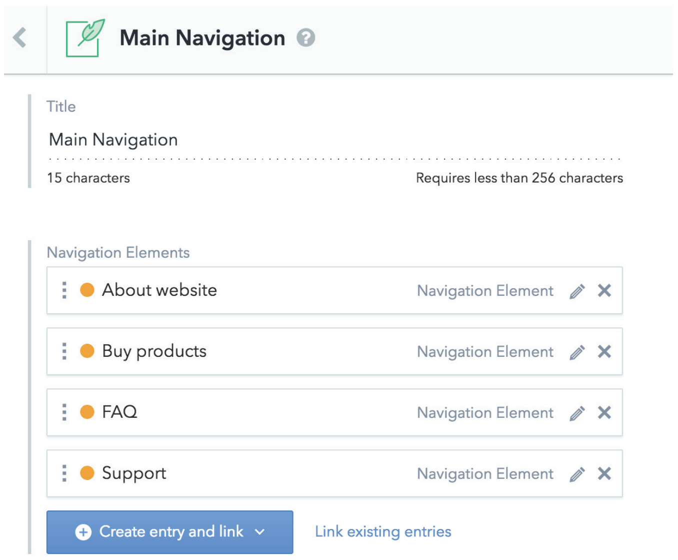
One way of thinking about navigation is to flip the above approaches on their head and create a content type specifically to hold navigation information. You're then able to structure pages or topics as references within this content type.

This allows for multiple navigation elements that can be especially useful if the website or application has different menus. This can be set up as a flexible or inflexible hierarchy depending on the fields and validation you include in your navigation content type. These can then be used as sub-assemblies within the content model and re-used or shown in different parts of the page.

An extension to the above approach is to make use of another content type, commonly called a routing object, to hold information about the location of the page (url part/screen depth etc.). This then can be attached either to the page assembly or topic, or used as the leaf node in place of the page or topic itself. This approach is useful for handling navigation separately.

Here is what you need to consider when choosing navigation content types:

- This navigation model is best for when you want fine control. As neither pages nor content define the relationships, you'll be able to choose the navigation to be displayed at a certain time and in a certain way.
- This model might throw you off at first because it flies in the face of good information architecture, not at all following typical rules.
- A good example of this way of navigating is with how a clothing retailer's website is set up. You'll have both men's and women's clothes, with the types of clothes being topics. You will also often find that these sites have "sales", "brands" and "new in" pages. The navigation element will encompass the categories (pages), and the clothing (topics) and then have these special pages. There is no hierarchy, just multiple ways of categorizing and accessing the same thing.
- You can use a hybrid of topics and assemblies with navigation content types. This will make navigation more flexible but might suit a more mature content strategy as things can get complicated.



Main Navigation

Title

Main Navigation

15 characters — Requires less than 256 characters

Navigation Elements

About website	Navigation Element
Buy products	Navigation Element
FAQ	Navigation Element
Support	Navigation Element

An example of navigation content types in Contentful

Being forced to understand your content and its relationships will ultimately help you make a better product, or at least you'll be communicating the information architecture more effectively through the application because you've managed to model it in a sensible way.

Ultimately if you want better navigation, consider putting thought into how the relationships between content areas should be defined and at which level you'll be better off. Contentful supports content models so flexible that it can be daunting when approaching something such as navigation, that has traditionally been dictated by a particular traditional CMS or just managed in markup. Just remember to start small, work collaboratively and you'll be fine.