

FALL 2021

EN

FR



# Tackling the Challenge of Content Localization

Contentful and Apply Digital Share Their Learnings



# Contents

## 02

### Tackling the Challenge of Content Localization

Executive Summary and Foreword

Why localization is vital for complex CMS implementations

The challenges of localized content

ISO codes for countries:  
Contentful's built-in locale strategy

## 11

### Approach 01 Page content localization

Related complexities

1. An author's nightmare
2. Single URL, multiple countries

## 13

### Approach 02 Page wrapper localization

Related complexities

1. An author's nightmare, continued
2. Performance issues

## 15

### Approach 03 Hybrid localization

Related complexities

1. Default language content required for Required fields
2. Unique slug for similar pages among countries

## 25

### Learnings

## 27

### Final thoughts

About Contentful

About Apply Digital

Credits





EN

FR

EN

ES

# Tackling the Challenge of Content Localization



# Executive Summary and Foreword

Today, most businesses are part of a global community. Their customer base is no longer limited by geography (and even if it is, that base is incredibly diverse and multifaceted by language and culture), with growing digital trends paving the way.

These diverse customers and the internal team members serving them expect access to education, information, assurances, support, and help for a variety of products and services that are on-demand and available in their native languages.

As tech professionals, we're tasked with building these solutions and services. So, we need to be equipped with tools powerful enough to traverse global digital landscapes and provide users with unique, tailored experiences. We can deliver on such diverse customer expectations through localization.

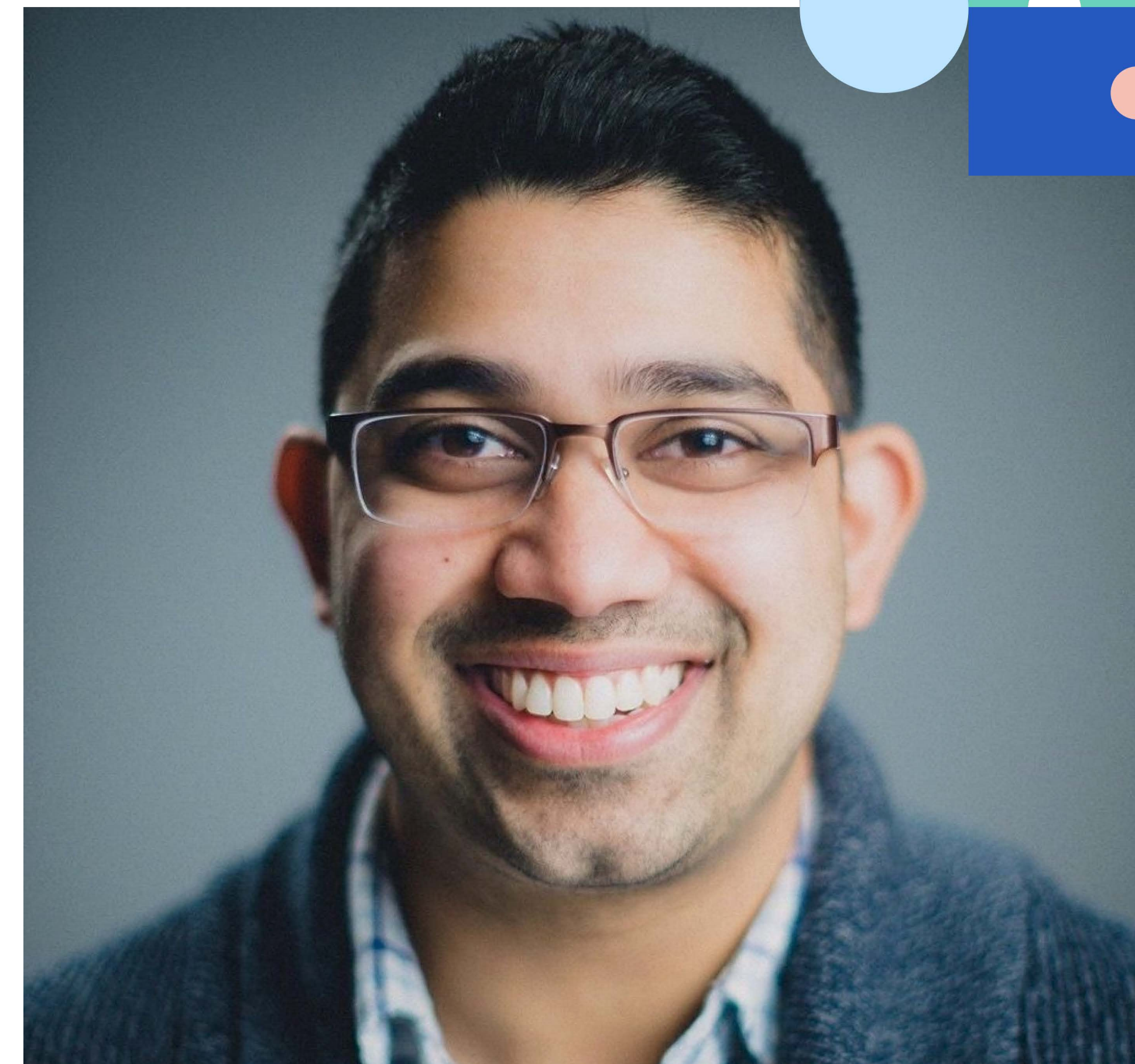
When a product is designed to serve diversity locally and globally, localization plays a critical role in its development and delivery. Localized products fit regional market conditions, eliminating language and currency barriers. In the competitive sense, localized products lower the burden to entry and customer friction, communicate brand integrity, increase customer loyalty, reduce churn, and improve sales.

This ebook illustrates how a strong partnership between Apply Digital and Contentful empowers businesses to tackle complex content localizations, translations, and personalization with incredible "digital-fast" speed. It also contains resources on how to execute localization and translations with Contentful effectively.

We hope it's useful to you in your own customer-led digital journey.

## Andrew Kumar

DIRECTOR, PLATFORM STRATEGY, CONTENTFUL





# Why localization is vital for complex CMS implementations

## Localizing content can be effortlessly achieved with the right content platform.

Localizing content can be effortlessly achieved with the right content platform. [Contentful's](#) convenient [built-in field-level localization](#) uses one-to-one translations within a multilingual region to support these efforts. However, the build can get more complicated if it involves multiple countries with unique content and language needs.

Let's look at an example of localization in action. A brand digitally marketing their product in Canada might need a homepage having a Hero Banner and a Card with content localized for English and French. If the same brand expands to Belgium, the homepage could require a Hero Banner and two Cards with content localized for Dutch, German, and French. It's times like these when our developers at Apply Digital pitch in to respond to dynamically complex requirements.

Our team recently launched a platform for a healthcare client using Contentful as their Headless content platform and Next.js for server-side rendering. We created a solution to support country-specific content and one-to-one translations within those countries. After diving deep into the challenges of localized content, we proposed several approaches tailored to meet our client's larger business needs before executing a successful solution.





# The challenges of localized content

While localization is essential for global customers (content authors) and end-users (those browsing the site), content centralization is equally important. It ensures everything is uniform and compliant with federal regulations.

Many legacy platforms achieve content localization through platform decentralization. This approach can leave end-users with loopholes, redirecting them to different websites. Imagine how disconnected the user experience would be if every website looked and felt different — whether it be a different logo or a separate set of content. This can and does occur when global organizations don't update company-specific sites at the same time or in the same way. What about when some content is common between all countries? There's still room for error. The varied language translation regulations in each country can complicate things.

Our team didn't take the decentralized route. Instead, we utilized Contentful's built-in locale strategy and came up with three approaches to centrally localize content on the platform.

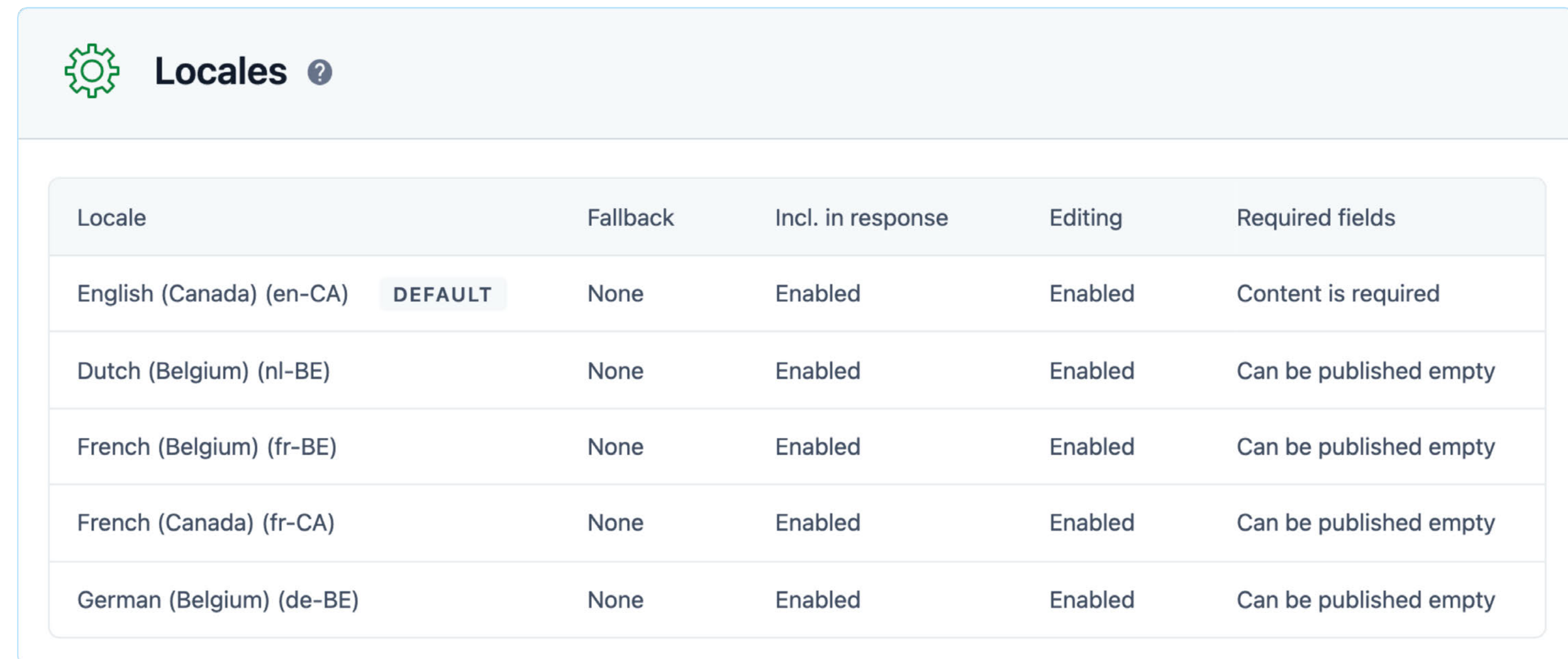
## Here's what we'll cover in the next section:

- ▶ ISO codes for countries:  
Contentful's built-in locale strategy
- ▶ Approach 1: Page content localization
- ▶ Approach 2: Page wrapper localization
- ▶ Approach 3: Hybrid localization



# ISO codes for countries: Contentful's built-in locale strategy

Our implementation began with oversimplifying the [Contentful data model](#) breakdown considering our project requirements.



Locale	Fallback	Incl. in response	Editing	Required fields
English (Canada) (en-CA) <b>DEFAULT</b>	None	Enabled	Enabled	Content is required
Dutch (Belgium) (nl-BE)	None	Enabled	Enabled	Can be published empty
French (Belgium) (fr-BE)	None	Enabled	Enabled	Can be published empty
French (Canada) (fr-CA)	None	Enabled	Enabled	Can be published empty
German (Belgium) (de-BE)	None	Enabled	Enabled	Can be published empty

We added all the required locales by jumping into Settings and selecting Locales.

**Page**

Fields (3)
JSON preview
Sidebar
Entry editors

⋮	<span style="border: 1px solid #000; border-radius: 50%; padding: 2px;">Ab</span>	<b>Entry Title</b> Short text	Entry title	<a href="#">Settings</a> ...
⋮	<span style="border: 1px solid #000; border-radius: 50%; padding: 2px;">Ab</span>	<b>Slug</b> Short text		<a href="#">Settings</a> ...
⋮		<b>Content</b> References, many		<a href="#">Settings</a> ...

On a given page, the Slug field has Appearance selected as Slug with Unique validation that auto-generates a path. The Content field is a one-to-many Reference that accepts only Hero Banners and Card entries as content types.


**Hero Banner**

Fields (3)
JSON preview
Sidebar
Entry editors

⋮	<span style="border: 1px solid #000; border-radius: 50%; padding: 2px;">Ab</span>	<b>Entry Title</b> Short text	Entry title	<a href="#">Settings</a> ...
⋮	<span style="border: 1px solid #000; border-radius: 50%; padding: 2px;">Ab</span>	<b>Title</b> Short text		<a href="#">Settings</a> ...
⋮		<b>Description</b> Rich text		<a href="#">Settings</a> ...

The Title is both Localized and a Required field for the two content types. For a Hero Banner, the Description field is also Localized and Required.




 **Card**

Fields (3)
JSON preview
Sidebar
Entry editors

⋮	Ab	<b>Entry Title</b> Short text		Entry title	<a href="#">Settings</a>	⋮
⋮	Ab	<b>Title</b> Short text				
⋮	Ab	<b>Call To Action</b> Reference				

The Call to Action field for a Card is a one-to-one Reference that accepts a Link entry type.

 **Link**

Fields (3)
JSON preview
Sidebar
Entry editors

⋮	Ab	<b>Entry Title</b> Short text		Entry title	<a href="#">Settings</a>	⋮
⋮	Ab	<b>Label</b> Short text				
⋮	Ab	<b>URL</b> Short text				

Label and URL are both Localized fields and Required for a Link type.

## After setting up our front-end, we carried out the implementation through Next.js' server-side rendering.

When a user lands on the Index page (Slug: /), our Next.js setup requests the corresponding Page Model from Contentful using the locale in the request object.

```
// next.config.js
module.exports = {
  i18n: {
    locales: ['en-CA', 'fr-CA', 'nl-BE', 'fr-BE', 'de-BE'],
    defaultLocale: 'en-CA'
  }
};
```

Configuration for Next.js that enables the out-of-the-box solution for internationalization to support the locales that we included in Contentful.

```
//utils/componentMapper.tsx
import { ExoticComponent, Fragment, ReactElement, ReactNode } from 'react';
import { HeroBanner } from '../components/HeroBanner';
import { Card } from '../components/Card';

interface IMapping {
  [name: string]: ({ content }: Page) => ReactElement;
}
```

The Page data is then parsed, and the Content field is passed down to the Component Mapper, which renders each component.



```
// Content Type ID : Custom Component
const mapping = {
  heroBanner: HeroBanner,
  card: Card
};

export const componentMapper = (
  contentType: string,
  componentMapping: IMapping = mapping
):
  | (({ content }: Page) => ReactElement) /* Function Component type */
  | ExoticComponent<{ children?: ReactNode }> /* Fragment type */ =>
  componentMapping[contentType] || Fragment;

//[...slug].tsx
export default function Page({
  page
}: InferGetServerSidePropsType<typeof getServerSideProps>): JSX.Element {

  return (
    <div>
      {page.content.fields.map(({ fields, sys }, index) => {
        const Template = componentMapper(sys.contentType.sys.id);
        return (
          <section key={`-${sys.id}-${index}`}>
            <Template {...fields} />
          </section>
        );
      })}
    );
  }
}
```

Now, content can be translated into any country-specific language. Our next goal was to allow separate content for every country — which required exploring a variety of solutions.



APPROACH 01

# Page content localization

01



# Page content localization

Allowing localization for the Content field of every [Page Model](#) enabled us to input custom content for every locale as per our requirements.

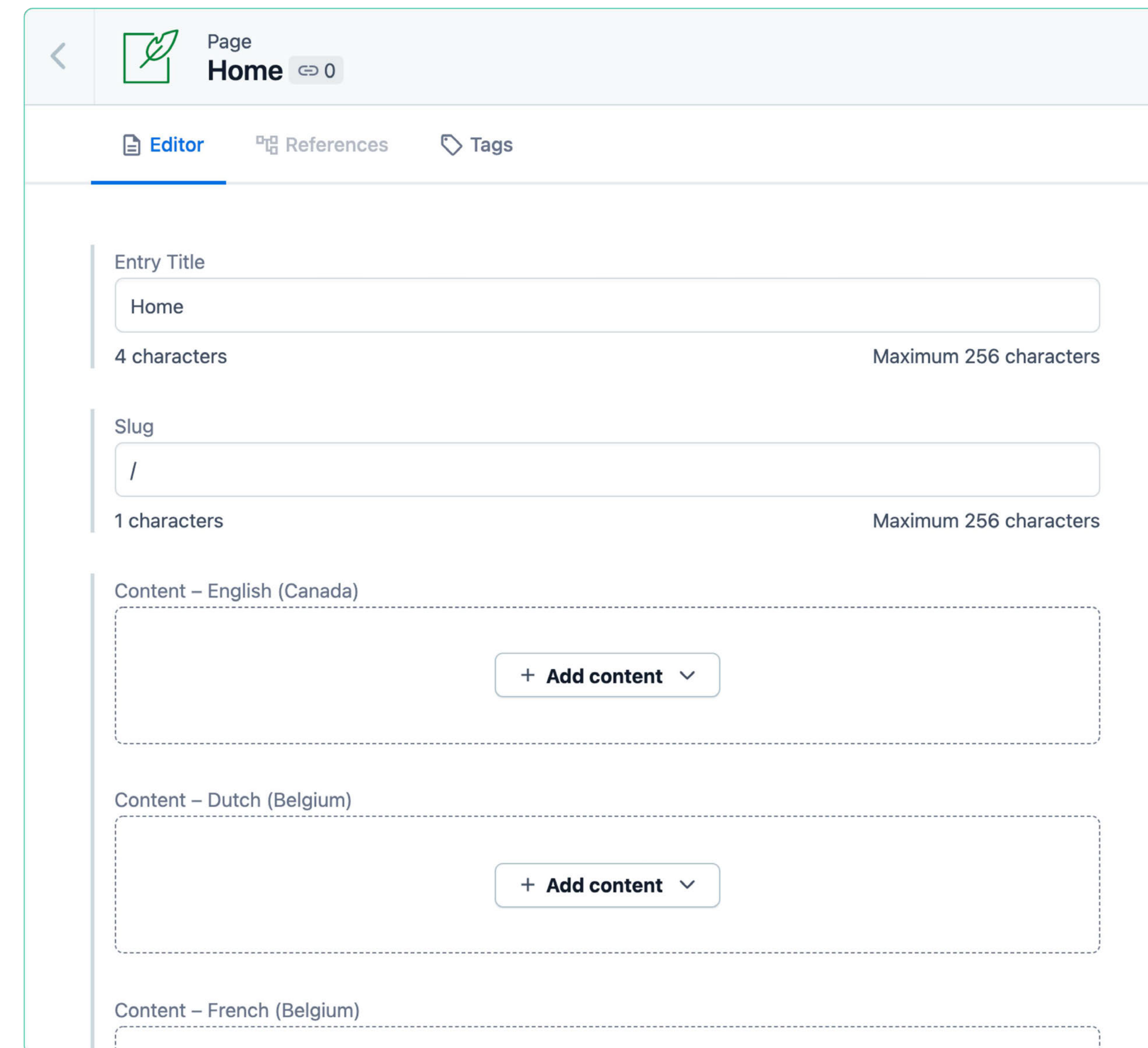
## Related complexities

### 1. An author's nightmare

Choosing this solution meant that authors would need to add content separately for all the languages within a locale. For example, a change of content in Canada would mean changing the content individually in pages created or translated in French (Canada) and English (Canada).

### 2. One single URL for multiple countries

This approach increased the workload of content authors — it also prevented us from generating separate URLs for different locales, which was crucial.



The screenshot displays a content management system interface for editing a page titled "Home". The page is currently in the "Editor" view, with tabs for "References" and "Tags". The "Entry Title" field contains "Home" (4 characters, maximum 256 characters). The "Slug" field contains "/" (1 character, maximum 256 characters). Below these fields, there are three localized content areas, each with a "+ Add content" button and a dropdown arrow:

- Content – English (Canada)
- Content – Dutch (Belgium)
- Content – French (Belgium)



APPROACH 02

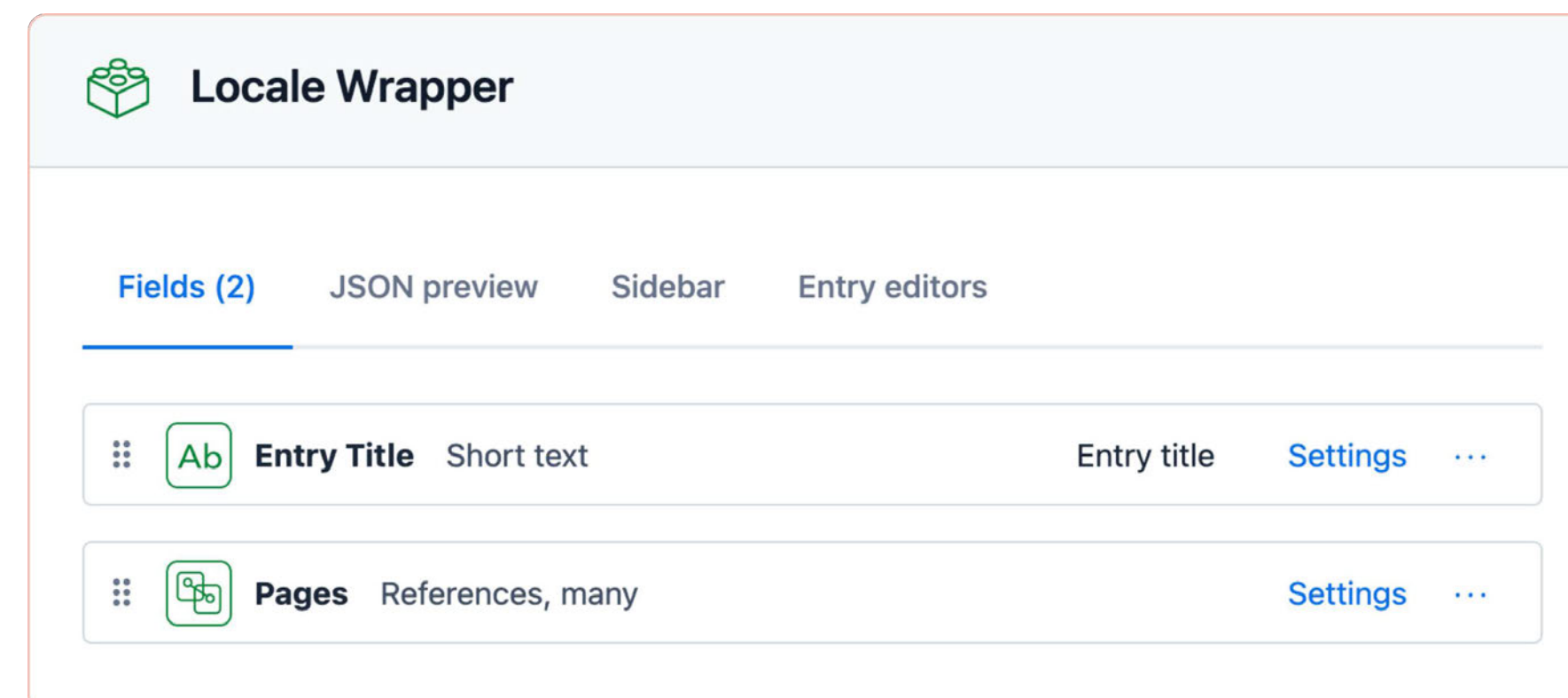
# Page wrapper localization

# 02

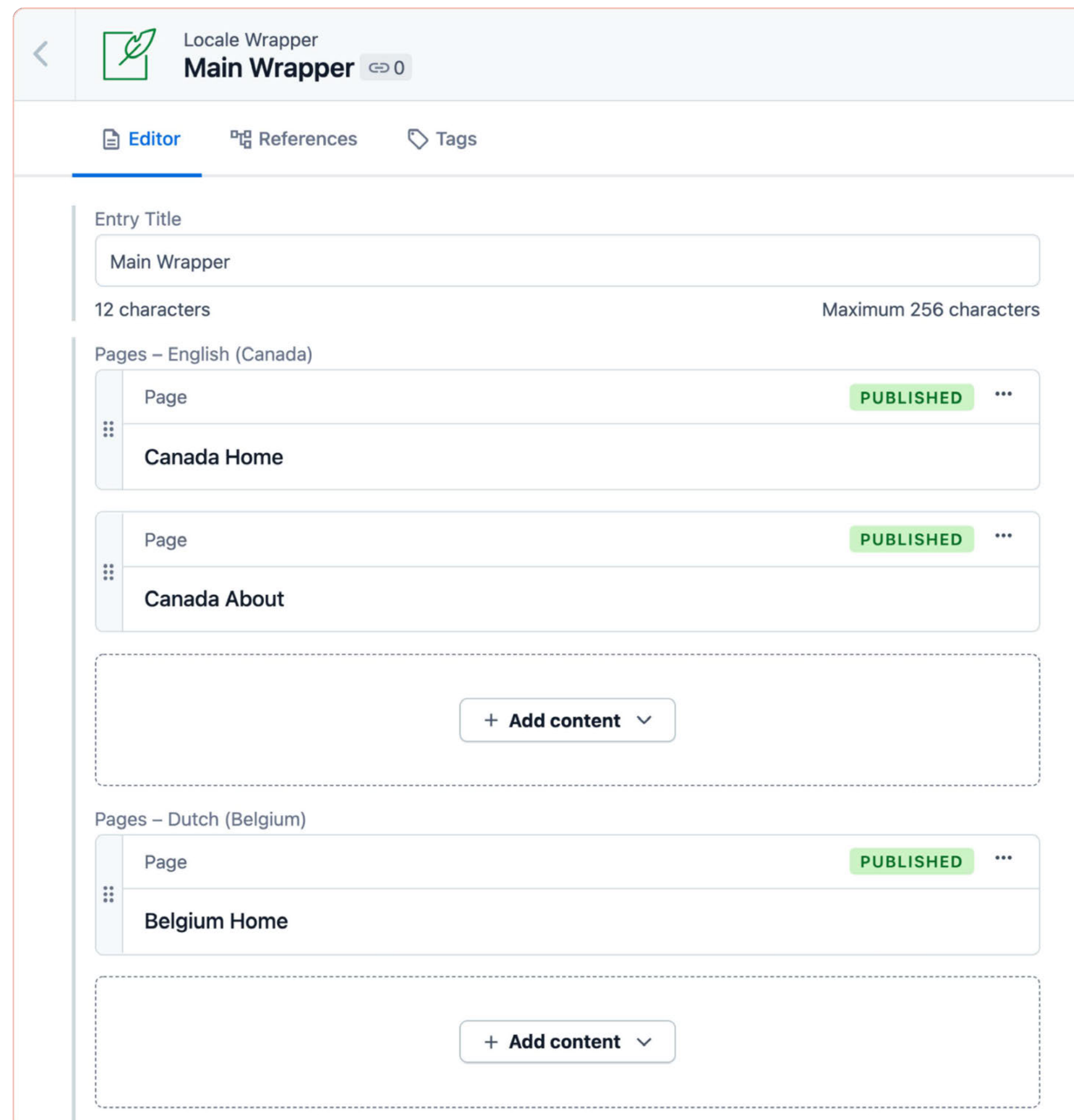


# Page wrapper localization

In this approach, we used a **Locale Wrapper** to wrap the **Page Model**, which allowed us to create all pages for a country within an individual wrapper.



We achieved this by making a Page field localized that accepts a one-to-one page-only Reference.



Every page is now linked to its locale. Each page also has its own URL and country-specific content, which brought us closer to fulfilling the client's requirements. Still, this came with its own set of complications.

## Related complexities

### 1. An author's nightmare, continued

Authors would still struggle because pages have to be separately linked for all languages within a locale.

### 2. Performance issues

A larger issue arose concerning the system's long-term performance when the number of pages grows. A single fetch returns all pages within a Locale Wrapper, meaning larger response times are inevitable. While Contentful allows us to filter requests by a field in the entry we request (`fields.slug=/`, i.e., a page in our case), it does not let us filter anything within a reference that is past the first level of the object (`fields.content.fields.slug=/`, i.e., the content on a given page). We would need to filter all the pages within a given locale.

Because we recognize the importance of performance and user satisfaction, we let go of this approach. Still, both the first and second approaches were important as they paved the way for our final approach, which remains in production today.



APPROACH 03

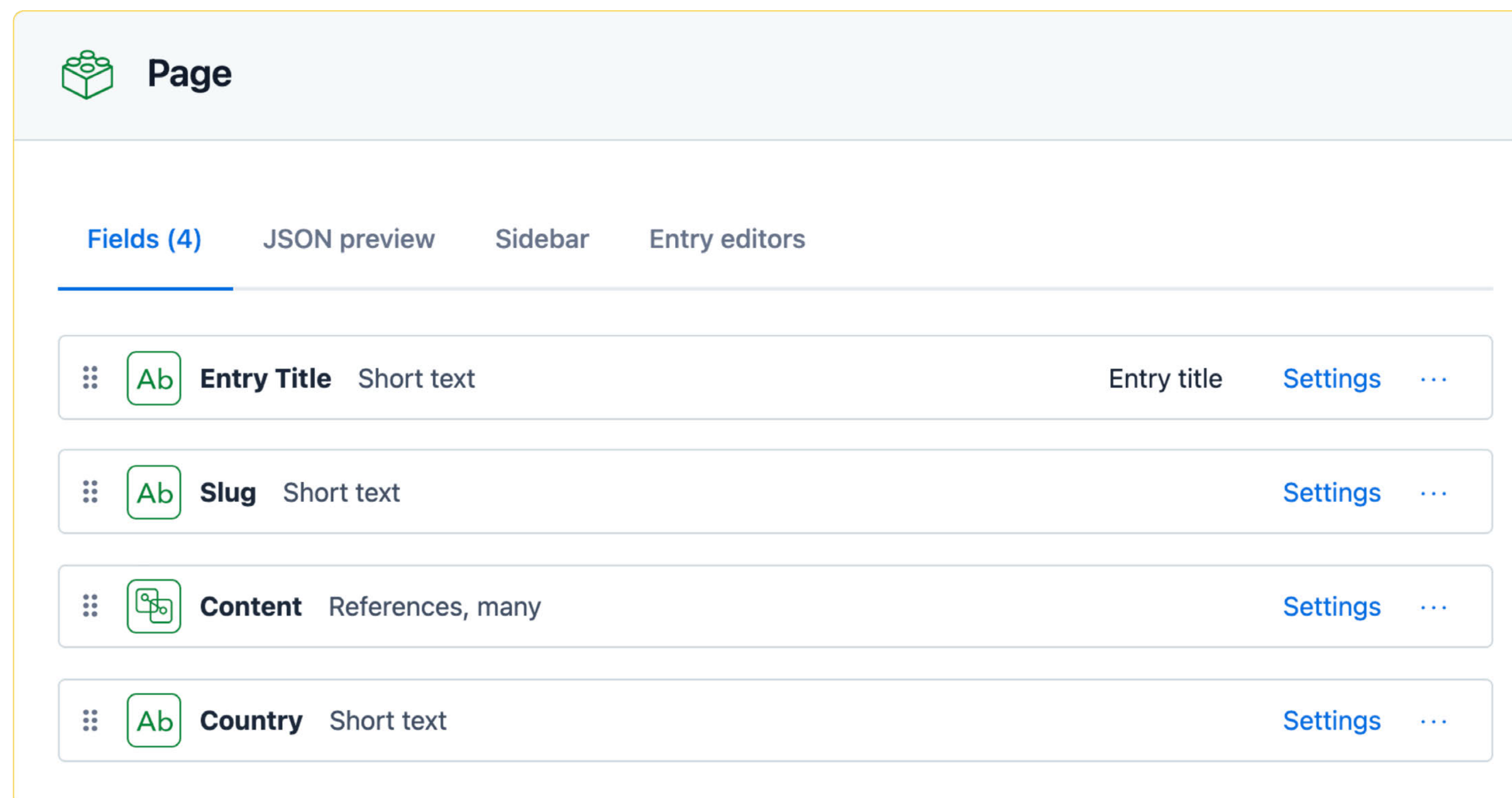
# Hybrid localization

# 03

# Hybrid localization

**This straightforward (and winning) approach includes the addition of a Country field within our Page Model.**

It is a Required field and provides options for country codes using a dropdown menu — for example, CA for Canada and BE for Belgium.



The screenshot displays the 'Page' model configuration in Contentful. The interface includes a header with a 'Page' icon and title, and a navigation bar with tabs for 'Fields (4)', 'JSON preview', 'Sidebar', and 'Entry editors'. The 'Fields (4)' tab is active, showing a list of four fields:

- Entry Title** (Short text): Entry title, Settings, ...
- Slug** (Short text): Settings, ...
- Content** (References, many): Settings, ...
- Country** (Short text): Settings, ...

We can now request a single page from Contentful that is filtered by country and page at once. This makes it easier for authors to create new country pages — they simply have to select the country of choice. To help with translation, they can also fill out the languages associated with each country.



## Related complexities

### 1. Default language content required for Required fields

In Contentful, a Required field — for example, a Hero Banner — must have the Title for the default language (English (Canada) in our case) filled out even when the country of reference doesn't require it. So, authors for Belgium would need to fill out the Title for English (Canada), which doesn't align with our updated requirements.

Hero Banner

Hero Banner 0

Editor References Tags

Entry Title

Hero Banner

16 characters Maximum 256 characters

Title (required) – English (Canada)

This is the title

17 characters Maximum 256 characters

Title – Dutch (Belgium)

0 characters Maximum 256 characters

Title – French (Belgium)

0 characters Maximum 256 characters

Title – French (Canada)

This is the title in FRENCH

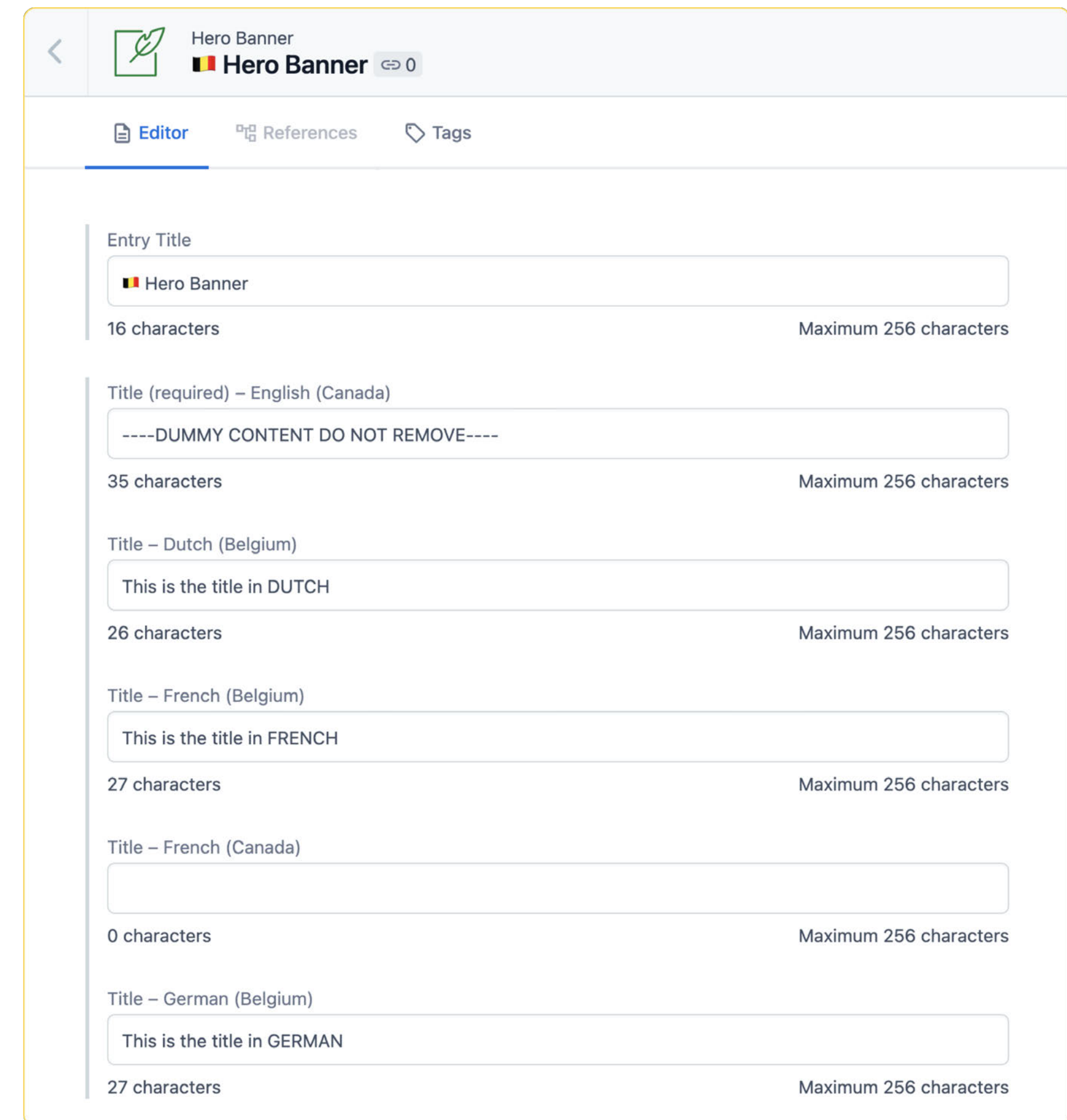
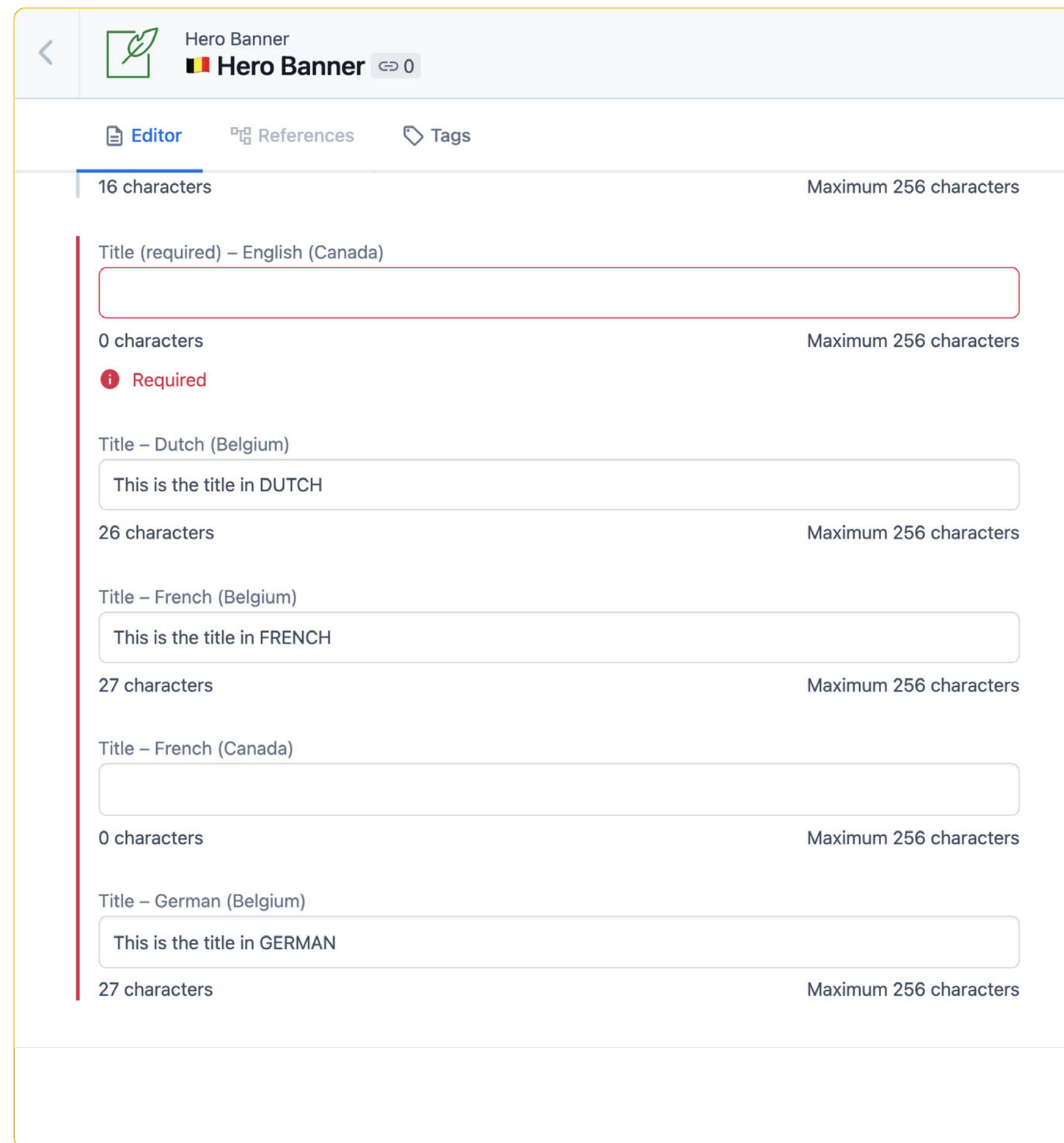
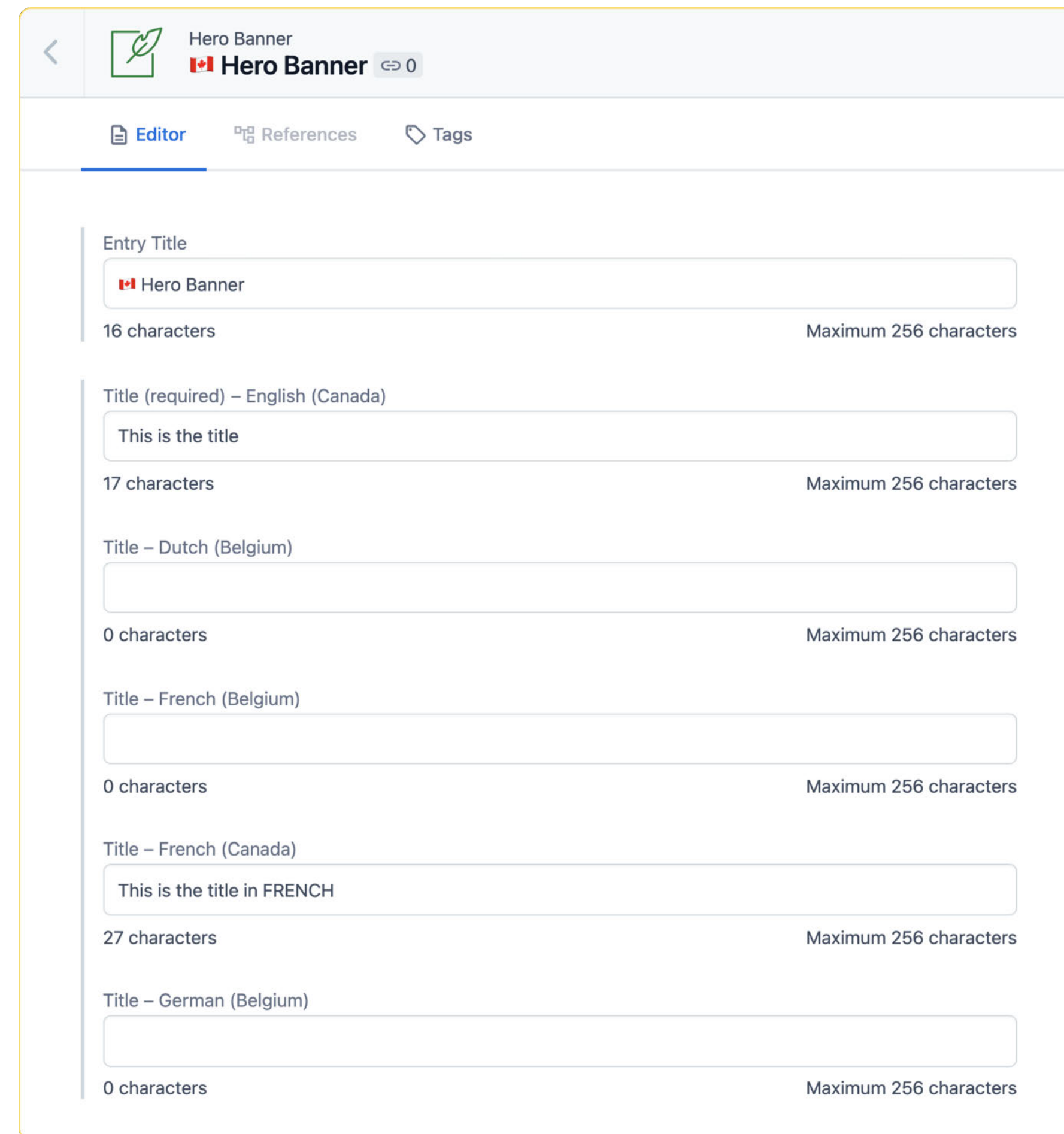
27 characters Maximum 256 characters

Title – German (Belgium)

0 characters Maximum 256 characters

To navigate this complexity, our developers explored two options:

### Dummy content for default language



For this, authors would simply need to add dummy content for English (Canada).

### Referenced content for default language

**Hero Banner**

Fields (3) | JSON preview | Sidebar | Entry editors

- Entry Title** Short text | Entry title | Settings ...
- Title** Short text | Settings ...
- Description** Rich text | Settings ...

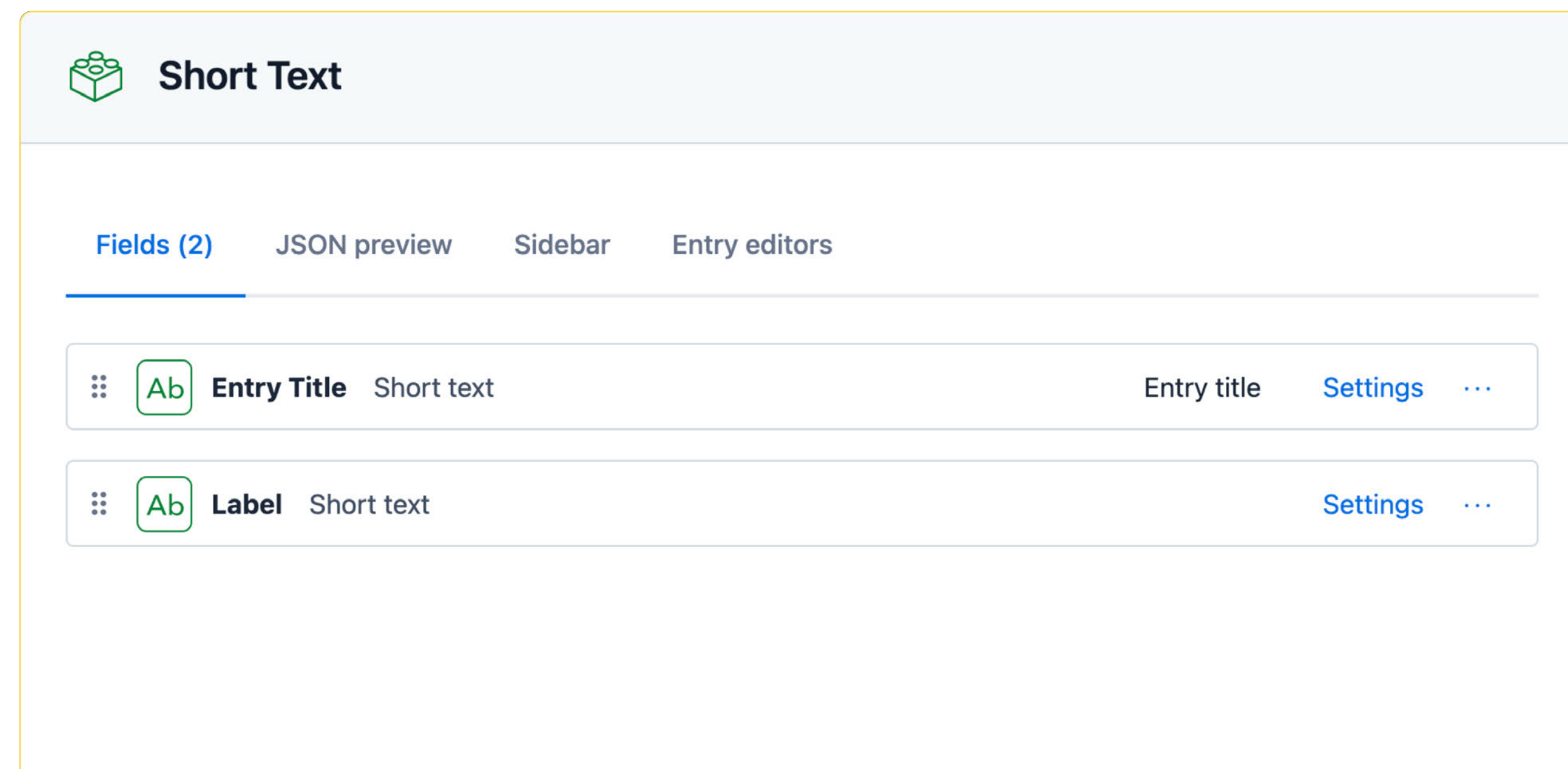
**Hero Banner**

Fields (3) | JSON preview | Sidebar | Entry editors

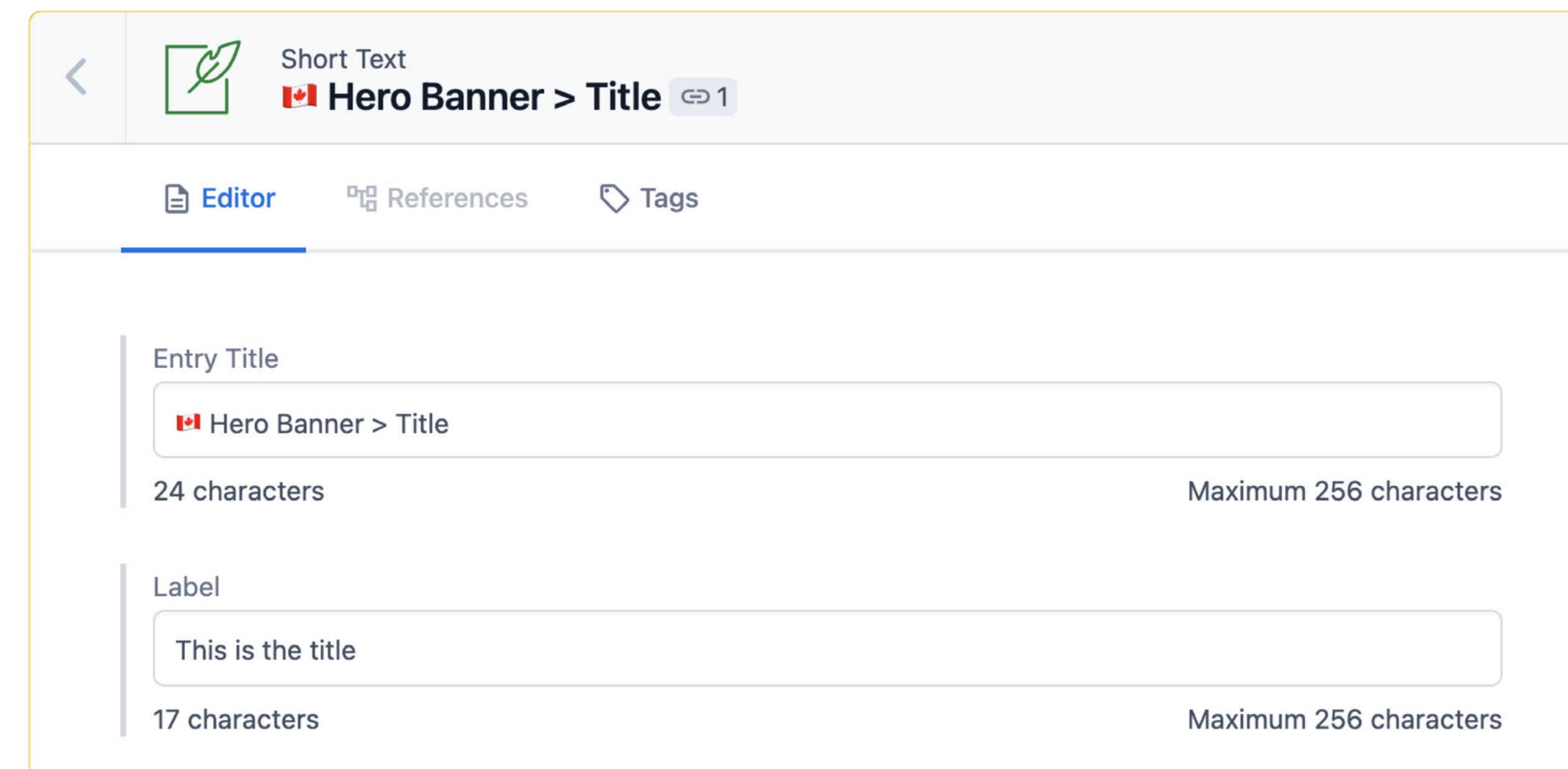
- Entry Title** Short text | Entry title | Settings ...
- Title** Reference | Settings ...
- Description** Rich text | Settings ...

We opted to change the Required field type from Short Text to Reference.



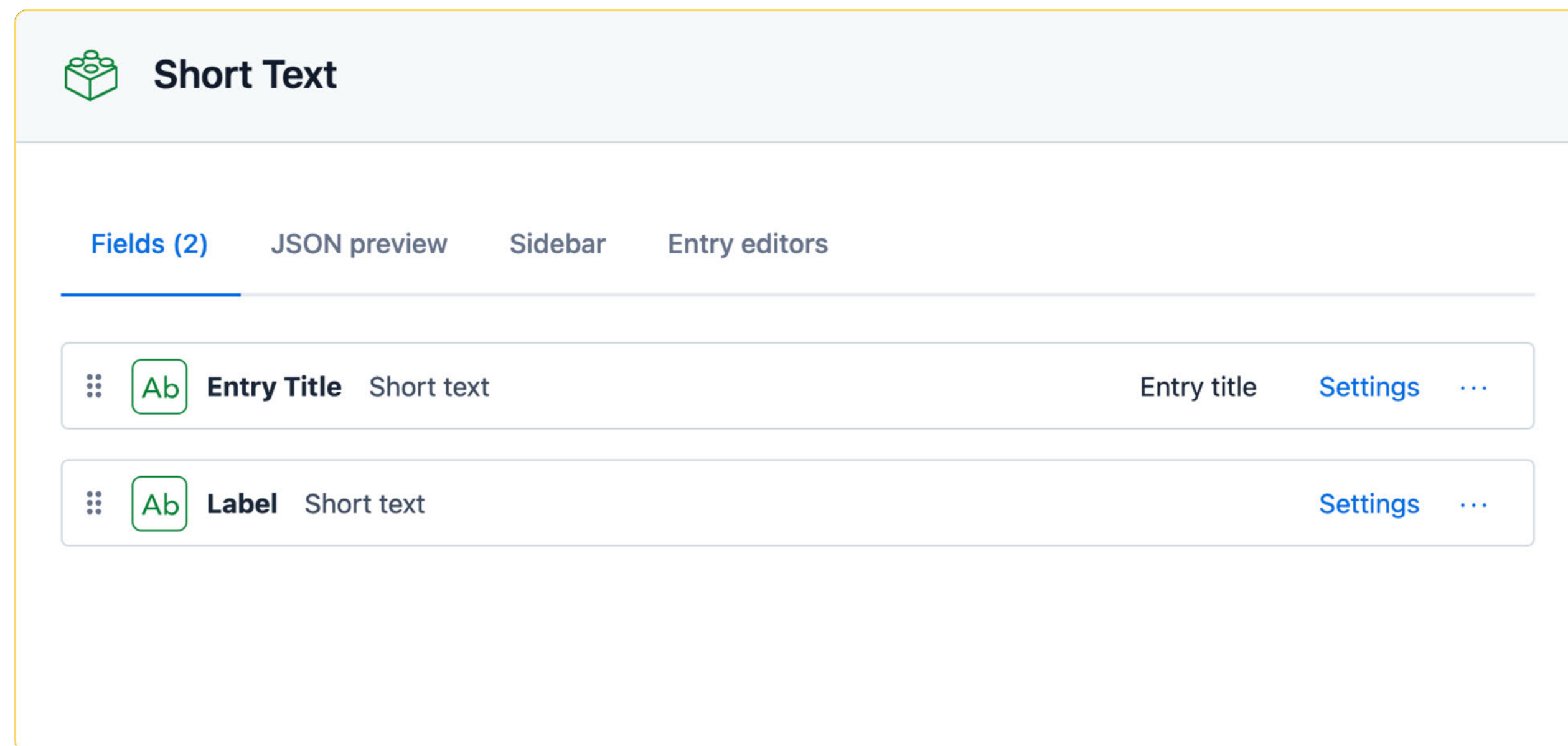


Then, we created a new Content Model for the Short Text field (with fields Entry Title as Text and Label as Short Text, which is Localized but not Required).

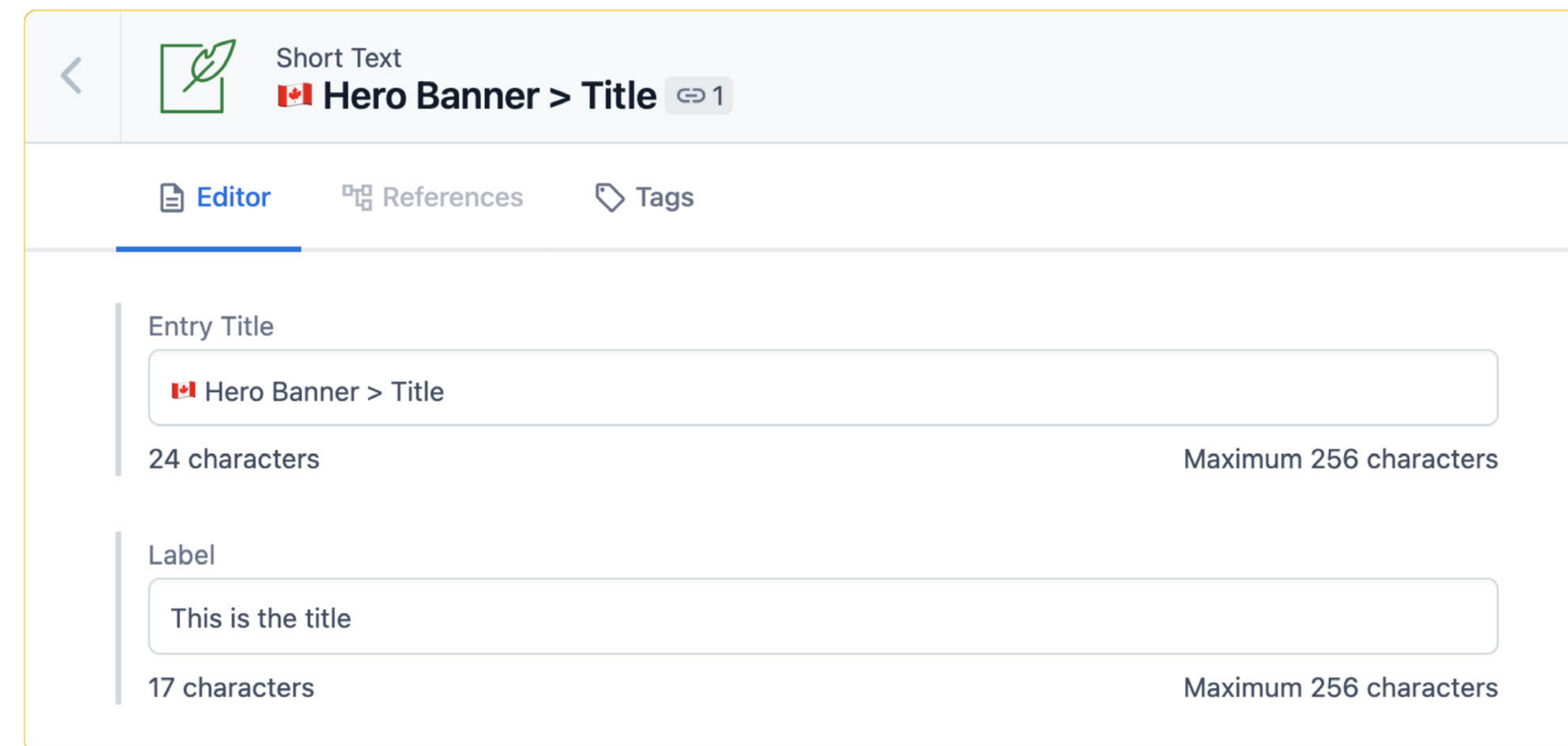


This Content Model could then be referenced from within the Required field.






Then, we created a new Content Model for the Short Text field (with fields Entry Title as Text and Label as Short Text, which is Localized but not Required).




This Content Model could then be referenced from within the Required field.

Hero Banner  Hero Banner 0

Editor References Tags


Entry Title

 Hero Banner

16 characters Maximum 256 characters

Title


Short Text PUBLISHED ...

 Hero Banner > Title

Description (required) – English (Canada)


Normal Text + Embed

This is the description

Hero Banner  Hero Banner 0

Editor References Tags


Entry Title

 Hero Banner

16 characters Maximum 256 characters

Title

Short Text PUBLISHED ...

 Hero Banner > Title

Description (required) – English (Canada)

Normal Text + Embed

----DUMMY CONTENT DO NOT REMOVE----

Description – Dutch (Belgium)

Normal Text + Embed

This is the description in DUTCH



Short Text **Hero Banner > Title** 1

Editor References Tags

Entry Title  
  
 24 characters Maximum 256 characters

Label – English (Canada)  
  
 17 characters Maximum 256 characters

Label – Dutch (Belgium)  
  
 0 characters Maximum 256 characters

Label – French (Belgium)  
  
 27 characters Maximum 256 characters

Label – French (Canada)  
  
 0 characters Maximum 256 characters

Label – German (Belgium)  
  
 0 characters Maximum 256 characters

Hero Banner **Hero Banner > Title** 0

Editor References Tags

Entry Title  
  
 24 characters Maximum 256 characters

Label – English (Canada)  
  
 0 characters Maximum 256 characters

Label – Dutch (Belgium)  
  
 26 characters Maximum 256 characters

Label – French (Belgium)  
  
 0 characters Maximum 256 characters

Label – French (Canada)  
  
 27 characters Maximum 256 characters

Label – German (Belgium)  
  
 27 characters Maximum 256 characters

Both solutions were effective workarounds. While our developers prefer the second option, the first is more popular among our customers, who are the main content authors.

## Related complexities

### 2. Unique slug for similar pages among countries

```
export const getCountryCode = (localeString: string): string => {  
  return localeString.split('-')[1];  
};
```

As a final step, we added a small function that fetches the country code from the locale (i.e., en-CA=CA). We passed this to Contentful along with the Slug and Locale to fetch the content requested by the user.

We initially set up the Slug field in pages as Short Text with Appearance as Slug with Validation Unique Field enabled. As we could have pages with the same Slug ("/") in different countries, this wasn't the right option for us.

We now use Short Text for the Slug with Appearance as Single Line with no Unique Validation and track them via spreadsheet for URLs per country.





EN

RU

ZH

AR

FR

ES

ID

PT

# Learnings



# Learnings

## Using Contentful as our Headless content platform has many benefits.

With handy ISO codes and the freedom to be creative on the front and back ends, our developers can easily incorporate localization. This made and continues to make a huge difference in our clients' end products.

There's always room for further growth, however. In the future, developers can look into:

### #1

Implementing workarounds to allow Required fields to have one-to-one translations rather than just the Default Locale

### #2

Experimenting with options to utilize the Appearance Slug without Unique Field Validations





# Final Thoughts



# Final thoughts

## Why should companies prioritize localization?

A localization strategy is expensive to retrofit. It's a foundational requirement that saves resources if architected at the beginning of the development process instead of being slapped on once a website is up and running.

## How does localization benefit clients?

A central localization strategy strengthens client relationships and supports happy customers and users. It also empowers our customers to push global updates into their system quickly and efficiently without having to deal with changing language regulations or employing intermediate agencies to handle individual websites.

Stay tuned for more ebooks that simplify great solutions created by our team.

Visit [applydigital.com](https://applydigital.com) or reach out to us at [hello@applydigital.com](mailto:hello@applydigital.com) to discover how we can help you move digital-fast and deliver the experiences your customers want and need.



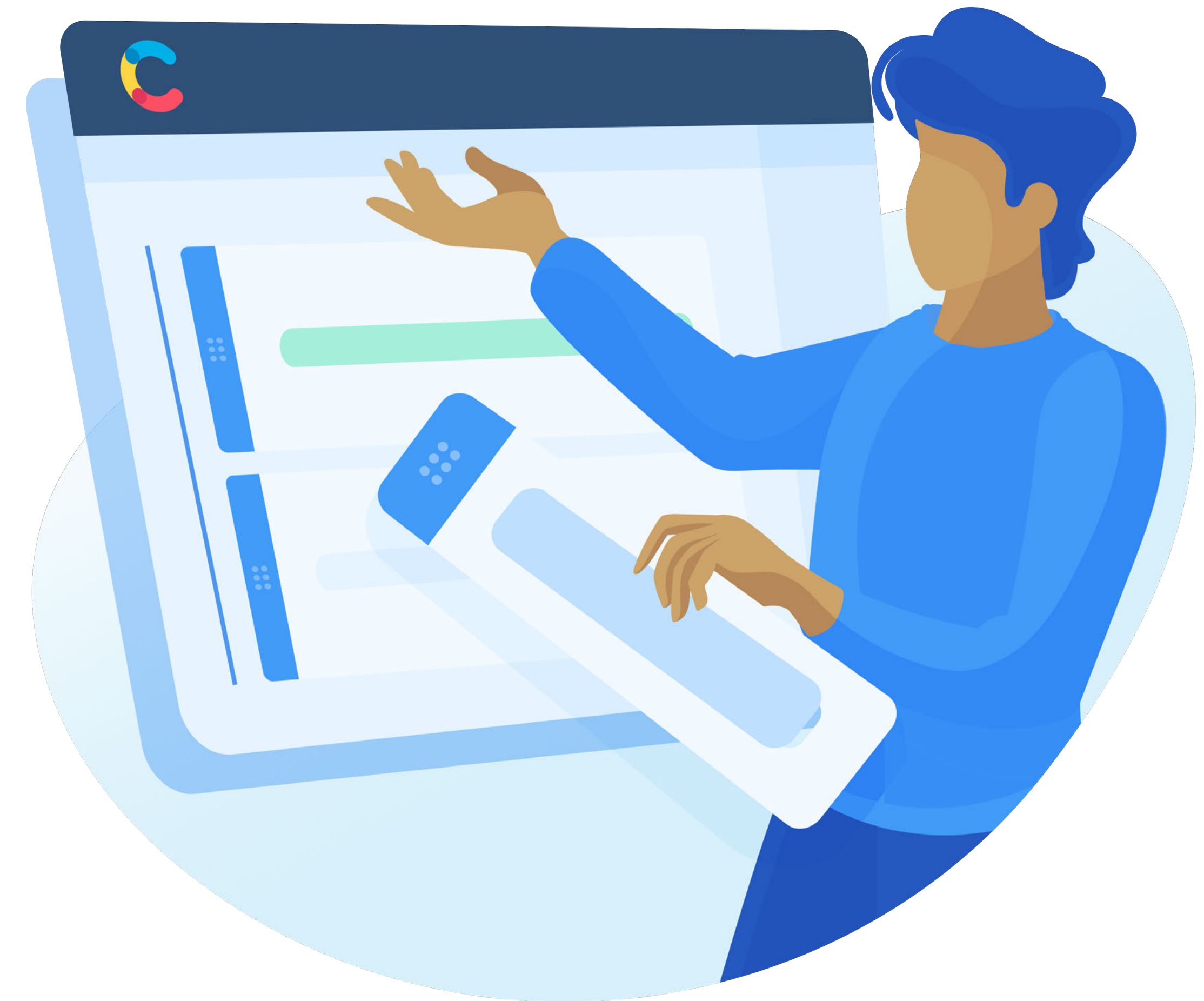


# About Contentful

Contentful, the leading content platform for digital-first businesses, helps 31% of Fortune 500 and thousands of brands around the world create and manage digital experiences for customers across any channel. The platform enables greater speed and scale than traditional CMS solutions.

Contentful unifies content in a single hub, structures it for use in any digital channel, and integrates seamlessly with hundreds of other tools through open APIs. Companies such as Chanel, Bang & Olufsen, Shiseido, Peloton, BP, and many others rely on Contentful.

For more information, visit [www.contentful.com](https://www.contentful.com)





# About Apply Digital

Apply Digital is an innovation, products, and experiences company.

Digital to our core, we are purpose-built to transform possibilities for people.

We solve complex problems with well-executed solutions tailor-made for continuous growth — we're ambitious and our clients are too. We work with well-funded start-ups, global brands, and Fortune 1000 companies spanning industries and audiences, including EA, Moderna, League Health, and Realtor.com.

**For more insights on how we can help you succeed,**

email us at [✉ hello@applydigital.com](mailto:hello@applydigital.com) or visit our website [🌐 www.applydigital.com](http://www.applydigital.com)





# Credits

**Gautam Lohia**

CEO

*Project Sponsor***Dale Shlass**

SENIOR DEVELOPER

*Lead Contributor***Gayan Pathirana**

HEAD OF TECHNOLOGY, EAST

*Subject Matter Expert***Andrew Kumar**

DIRECTOR, PLATFORM STRATEGY

*Subject Matter Expert***Rashika Srivastava**

WRITER

*Lead Writer and Researcher***Jaime Chang**

WRITER

*Copy Support***Liz Goode**

SENIOR WRITER

*Content Supervision***Daniela Valdes**

MARKETING DESIGNER

*Design Lead***Mauricio Pommella**

HEAD OF DESIGN

*Design Supervision***Vaibhav Lohia**

DIRECTOR OF MARKETING

*Delivery Lead***Yael Rubinoff**

PROJECT MANAGER

*Project Management Lead*