Get localized content items

February 28, 2023 • Jan Cerman • 4 min read • GraphQL

Localization in Kontent.ai allows you to define multiple languages for your project and create localized culture-specific content. Let's look at how you can retrieve localized content from your project.

Choose a URL pattern

When developing multilingual web apps, it is a good practice to ensure unique URL addresses for each page.

1. Language prefixes

One approach is to use language prefixes in URLs, such as `en-us` or `es-es`, to differentiate between content cultures.

For instance, the URLs can be `https://myapp.com/en-us/about-us` for English language and `https://myapp.com/es-es/about-us` for Spanish language. In this scenario, you get content items in a specific language.

2. Localized URL slugs

Another way is to combine the language prefixes with localized names of your content items. That is using localized URL slugs.

For example, a URL slug of the *About us* content item in English is `about-us`. Translated to Spanish, the URL slug changes to, for example, `acerda-de-nosotros`. The URL for the *About us* content item in Spanish becomes `https://myapp.com/es-es/acerda-de-nosotros`. In this case, you need to get content items by their URL slug value.

> If you're thinking through your URL patterns, make sure to check considerations for multilingual URLs.

Languages and localization

Each language in your Kontent.ai project is uniquely identified by its codename. This codename can be any string you choose, such as, `English`, `en-US`, or `en-GB`.

> **Your project's default language**
>
> Every project comes with a single default language that cannot be removed. When you request content without specifying a language, the Delivery API returns content in the default language.

Get localized content items

To get localized content, specify the `language` parameter and provide the codename of the requested language. For example, to get an article named *About Us* article in the Spanish language, specify the item's codename and the language's codename.

GraphQL

```
query GetArticleInSpanish {
  article(codename: "about_us", languageFilter: {languageCodename: "es-ES"}) {
    title
  }
}
```

If the content item doesn't exist in the specified language, the Delivery API checks if the content exists in the fallback language configured for the requested language.

Get items by localized URL slug

Depending on how you implement routing in your app, you might want to request content items based on the knowledge of their URL slugs.

To retrieve a content item by a localized URL slug, you need to specify the `language` parameter and then filter by URL slug value. If you omit the `language` parameter, you get content in the default language.

For example, to get an *About us* content item whose URL slug in Spanish is *acerda-de-nosotros*, you need to specify the following:

- The language's codename using the `language` parameter. For Spanish it's `es-ES`.
- The content item's URL slug value using the `equals` filter.

GraphQL

```
query GetArticlesByLocalizedSlug {
  article_All(where: {title: {eq: "acerda-de-nosotros"}}, languageFilter:
{languageCodename: "es-ES"}) {
    items {
      title
    }
  }
}
```

If the specified query doesn't match any content items, the Delivery API returns an empty response.

Ignoring language fallbacks

To ignore language fallbacks and retrieve content items only in the specified language, you need to filter out any content items whose language is different from the specified language.

The following table shows what content the Delivery API returns for specific combinations of language-related filters. The example works with a Kontent.ai project with the default language set as English `en-US` and the second language set as Spanish `es-ES`, which falls back to the default language.

- `language=es-ES` – Specifies the codename of the requested language.
- `system.language=es-ES` – Filters content items by their language.

`language` parameter value	`system.language` parameter value	Behavior
`es-ES`	<not set>	The API returns content items in Spanish. If an item is not translated to Spanish, the API returns the English (`en-us`) version of the items as fallback.
`es-ES`	`es-ES`	The API returns only content items in Spanish. If an item is not translated to Spanish, the API doesn't return it.
`es-ES`	`en-US`	The API returns only content items in English provided they don't have content in Spanish. If an item is translated to Spanish, it is not returned.

For example, this is how you retrieve content items translated to Spanish language without any fallbacks.

GraphQL

```
query GetArticlesInSpanishOnly {
  article_All(
    # Requests items in Spanish
    languageFilter: {languageCodename: "es-ES"},
    # Filters the items so that only Spanish is returned, ignoring language fallbacks
    where: {_system_: {language: {_system_: {codename: {eq: "es-ES"}}}}}) {
    items {
      title
    }

```

```
    }
}
```

What's next?

You've learned how you to handle URLs in multilingual apps and two ways of retrieving translated content. If you know the content item's codename, you can request it directly and specify a language. When you need to request an item by its localized URL slug, use filters to get the item you need.

- Check examples of filtering content via Delivery API.
- Integrate a translation management system to automate your translations with Kontent.ai.
- Read blog posts about automating translation with Kontent.ai.
- Set up content preview in your app and project so that content contributors can review localized content with confidence.