Set up preview for content items

April 11, 2023 • Jan Cerman and David Klement • 9 min read • JavaScript

Set up content preview for your Kontent.ai projects so that your content creators can preview unpublished content and review it with confidence.

In this article, you'll learn to:

- Teach your app to preview latest content.
- Set up your app so that it runs in two instances – production and preview.
- Leverage spaces for multi-regional or multi-brand projects.
- Define preview URLs for content types you want to preview.
- Prepare preview for Web Spotlight.

Learn to get your latest content

Start by teaching your app how to fetch the latest versions of your content. This means making requests to Delivery Preview API. Every request to the API must be authenticated with a Preview API key. Keep in mind that server-side technology is required to keep your API keys secret.

- Preview content with Delivery REST API
- Preview content with Delivery GraphQL API.

To get the Preview API key:

1. In Kontent.ai, go to **Environment settings** > **API keys**.
2. In the **Preview API** card, click for one of the keys.

Quick facts about the Primary and Secondary keys

- The scope of the API keys is per environment.
- Their default expiration date is 1 year.
- Use the Primary key for continuous use in your apps.
- Use the Secondary key when revoking the Primary key to prevent downtime.

Here's an example of how you can get the latest version of a *My article* content item:

JavaScript

```
// Tip: Find more about JS/TS SDKs at https://kontent.ai/learn/javascript
const KontentDelivery = require('@kontent-ai/delivery-sdk');

const deliveryClient = KontentDelivery.createDeliveryClient({
  environmentId: '<YOUR_ENVIRONMENT_ID>',
  previewApiKey: '<YOUR_PREVIEW_API_KEY>',
  defaultQueryConfig: {
```

```
    usePreviewMode: true, // Queries the Delivery Preview API.
  }
});

const response = await deliveryClient.item('my_article')
  .toPromise();
```

Prepare your app for preview

When your app can display the content it gets from a Kontent.ai project, the next step is to prepare your app for previewing unpublished content.

We recommend using Delivery SDKs to make all this easier.

To follow the best practices and avoid using your production app for fetching non-production data (preview), you need two separate instances of your app:

- Production instance: Fetches published content via Delivery API. This instance serves content via `example.org` URL, for example.
- Preview instance: Fetches the latest version of your content, even if it's not published yet, via Delivery Preview API. This instance serves content via `preview.example.org` URL, for example.

You can distinguish the instances by providing or omitting the Preview API key. If the key's present, your app will run the preview instance, if it's not, that's a sign to run the production instance.

Set up multiple previews with spaces

If you're working on a multi-regional or multi-brand project, you can use spaces to set up multiple previews for your websites.

Every environment can have several spaces. You can use different domains and preview URLs for each space.

1. In ⚙ **Environment settings**, select **Spaces**.
2. Click **Create new space**.
3. Type the new space name.
4. Click **Save**.

Create as many spaces as you need.

Once you're done, go to **Environment settings** > **Preview URLs** > **Space domains** and type in the domain for each space such as `preview.example.org`.

Define preview URLs for content types

To preview your content, you need to set up preview URLs for your content types. The URLs may be different for each combination of space and content type.

1. In **Environment settings** > **Preview URLs** > **Preview URLs for content types**, click **Set up preview for a content type**.
2. Select a content type.
3. In **Used in spaces**, select a space you're setting the URL for.
   - If you don't use spaces, select *All spaces*. Similarly, use *All spaces* for a preview URL that's universal across all your spaces.
4. Type in the preview URL.

The preview URLs must be:

- Absolute paths in the format `https://domain/path`
- Shorter than 1024 characters

> **Use a secure connection**
>
> Preview URLs require the `https://` protocol and URLs accessible to Kontent.ai. Without a valid SSL certificate, Kontent.ai responds with secure connection errors.
>
> When developing apps locally, you can serve pages over HTTPS in combination with ngrok's forwarded address.

Dynamic preview URLs with macros

If you're setting up URLs that need to be dynamic, there are a few macros you can use:

- `{Space}`: space domain
- `{URLslug}`: content item's URL slug element value
- `{Lang}`: codename of the selected language
- `{Codename}`: content item's codename
- `{ItemId}`: content item's internal ID
- `{Collection}`: content item's collection codename
  - This macro is available if you have collections enabled for your subscription.

You can create SEO-friendly URLs using the URL slug element with the `{URLslug}` macro: `https://preview.example.org/articles/{URLslug}`. Alternatively, you can use the `{Codename}` or `{ItemId}` macros to identify the content items in your preview URLs.

In multi-regional projects, use the `{Lang}` macro to get the currently selected language codename, such as `en-us` or `ja-jp`.

You can combine the macros. A preview URL `https://preview.example.org/{Lang}/articles/{URLslug}` would resolve to `https://preview.example.org/en-ca/articles/previewing-content-items`, for example.

Multiple content items can have the same URL slug

Kontent.ai doesn't check if a URL slug value is unique across your project. This means you can have multiple items with the same URL slug.

However, if you need unique URL slugs, you have several options to ensure that you don't end up with multiple same URL slugs. Check out your options to achieve unique URL slugs.

Static preview URLs

Your home page, for instance, doesn't need a dynamic preview URL, because it's static (it's unique and in one location).

For example, a home page's preview URL for an app running at `https://preview.example.org` would be `https://preview.example.org` or `https://{Space}`.

Set up a preview for Web Spotlight

When setting up Web Spotlight, you set up preview just like you would without Web Spotlight.

Because the preview URLs (that is your app) will be loaded and sandboxed in an `<iframe>` within Kontent.ai, you need to ensure a few things for your preview environment:

- Always use a secured connection (HTTPS) for your URLs, such as `https://preview.example.org`.
- Allow your app to load within Kontent.ai using the `Content-Security-Policy` header and the frame-ancestors directive: `frame-ancestors https://app.kontent.ai`.
  - To enable opening external links from your app, such as PDFs stored as assets in Kontent.ai, you need to set `sandbox` directives using the `Content-Security-Policy` header. For example, the directive `allow-popups-to-escape-sandbox` allows opening external links, `allow-forms` allows submitting forms, `allow-downloads` allows downloads after users click a link, and so on.
- Make sure your web browser allows third-party cookies and that your cookies are designed for a 3rd-party context by setting cookies attributes `SameSite=None; Secure`.
- If you first want to test your implementation locally, you need to generate a self-signed HTTPS certificate.

Web Spotlight and spaces

Web Spotlight uses a content item of the *Web Spotlight root* content type as its top-level (root) content item. The rest of the tree is under this item.

Setting up a preview for Web Spotlight with multiple spaces is no different from setting it up for Web Spotlight in a project with only a single space.

You set up the preview for the *Web Spotlight root* content type and for all the other types under it. Specify different preview URLs for different combinations of spaces and content types.

Set up edit buttons in your preview

Once you have a preview working in both your app and Kontent.ai project, it's a good idea to add edit buttons to your application. They can look like what you see below.

See the code example on https://codepen.io/Kontent-ai-Learn/pen/MWpXvQz

For content creators, this small addition means they can go straight to editing in Kontent.ai just by clicking the button. This will help them in case they need to fix minor errors or typos right

after they spot them when previewing their content.

If your editors have problems opening items through the edit links, verify their user roles.

What's next?

- Set up URL slugs in your content model so that content contributors know which elements to fill in.
- Know your options for setting up routing and URLs in your app.
- Get your items by URL slug in your app.
- Build your app right by following best practices on getting content.
- Take an in-depth look at the Delivery REST API endpoints.
- Content model changes (think content types or taxonomies) affect your production immediately. Use multiple project environments to test them before you deploy them into production.