Set up preview for content items

December 15, 2021 • Jan Cerman • 10 min read • JavaScript

Learn how to set up content preview for your Kontent by Kentico project. This involves adjusting your app so that it retrieves the latest content when needed. You'll set up environments for your app and make sure your app recognizes them. You'll also set up preview URLs in your project so that Kontent knows which pages to open for previewing specific content types and items.

At the end, you'll have preview URLs for each type of preview-able content, like articles or a home page. Doing all this lets your writers preview unpublished content and review it with confidence. Let's begin!

Get the latest version of everything

Start with teaching your app how to fetch the latest versions of your content. You do that by making requests to the Delivery Preview API. Every request to the API must be authenticated with a Preview API key. There are two keys, Primary and Secondary.

> **Primary vs. Secondary key**
>
> The following instructions work equally for both the Primary key and the Secondary key. Both API keys are generated per project and have no expiration date. For continuous use, we recommend using the Primary key. Use the Secondary key only when revoking the Primary key to prevent downtime.

To get the Preview API key:

1. In Kontent, select your project.
2. From the app menu, select Project settings.
3. Under **Environment settings**, select **API keys**.
4. In the **Preview API** box, click .

Use the key in your app to authenticate your requests to the preview versions of the Delivery REST API and Delivery GraphQL API. For example, this is how you can get the latest version of a *My article* item.

JavaScript

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

const deliveryClient = KontentDelivery.createDeliveryClient({
  projectId: '<YOUR_PROJECT_ID>',
  previewApiKey: '<YOUR_PREVIEW_API_KEY>',
  globalQueryConfig: {
    usePreviewMode: true, // Queries the Delivery Preview API.
  }
});


```

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

Set up environments for your app

Assuming your app can correctly display the content it gets from a Kontent project, your next step is ensuring that your app runs in two separate environments. Let's call them production and preview. In the production environment, your app gets and displays only the published content. In the preview environment, your app gets and displays the latest versions of your published and unpublished content.

The main difference between the two environments will be the environment variables available to your app. In the production environment, you might specify a variable like `SecureAccessKey` (if you're using secure access), which lets your app see if it's running in a production environment. In such a case, the app fetches published content via the Delivery API. For the preview environment, you might specify a variable like `PreviewApiKey`, which lets your app see if it's running in a preview environment. In this case, the app will fetch both published and unpublished content via the Delivery Preview API. When it comes to getting the content itself, both versions of the Delivery API work the same.

The way to check the environment variables depends on your tech stack and the service you use to deploy your app. For example, your application can check whether it's running in a preview environment, set a boolean flag based on the check, and use the Delivery (Preview) API as a result. With the Delivery SDKs, the API to use is usually determined by providing or omitting the Preview API key. Keep in mind that server-side technology is required to keep your API keys secret.

> **Need to update your project structure?**
>
> If you use only the production and preview deployment environments for your app, any content model change (think content types or taxonomies) to your project will affect both deployment environments. If you want to introduce a new model or change your existing model, we recommend using multiple project environments.

Set up content preview in your project

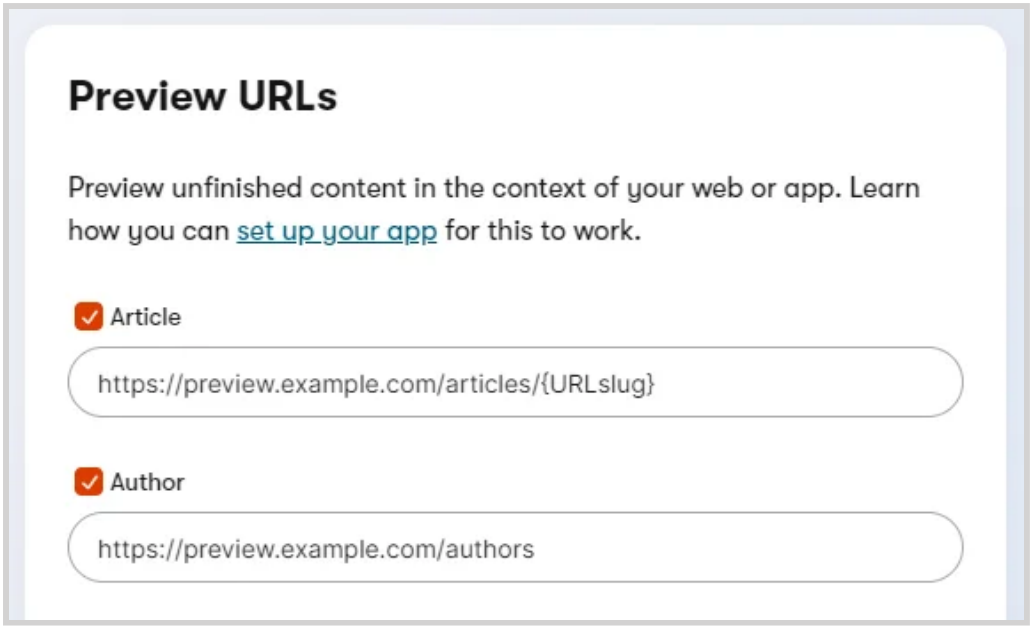
Once you've got your app running in a preview environment, you need to specify where (URL-wise) each type of your content can be accessed and viewed. For example, imagine your app runs at `https://preview.example.com` and you want to set up preview for articles and the home page.

Each article has its own URL identifier. In Kontent, this identifier is called a URL slug and there's a content type element of the same name. The URL slug element contains an SEO-friendly text usually generated from the article's title. For the home page, however, you don't need a URL slug because the home page sits at the root of your site.

To set up preview URLs for your project:

1. In Kontent, select **Project settings** from the app menu.
2. Under **Environment settings**, choose **Preview URLs**.
3. For each content type of preview-able content:
    1. Select the checkbox with the content type's name.

2. Type in the preview URL.



Let's take a closer look at that last step. The preview URLs you provide must be absolute and in this format `protocol://domain/path`.

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

Preview URL for articles

The URL slug of articles is dynamic (each article is titled differently) and you need to reflect that in the preview URL. To create dynamic preview URLs, you need to use macros.

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

Use the `{URLslug}` macro in your preview URLs like this `https://preview.example.com/articles/{URLslug}`. If you don't use URL slugs in your content items, you can use the `{Codename}` or `{ItemId}` macros to identify the content items in your preview URLs.

In multilingual projects, use the `{Lang}` macro in your preview URLs. This helps differentiate between languages. The `{Lang}` macro resolves to the currently selected language's codename, such as `en-us` or `es-es`, in your Kontent project.

All macros can be used together. For example, by combining the `{URLSlug}` and `{Lang}` macros you can create preview URLs such as `https://preview.example.com/{Lang}/articles/{URLslug}` that might resolve to `https://preview.example.com/es-es/articles/en-asados`.

> **Multiple content items can have the same URL slug**
>
> Kontent doesn't check if the URL slug values are unique across your project. This means you can have multiple items with the same URL slug.
>
> If you need to ensure uniqueness for your URLs, you have several options:
>
> - Use multiple identifiers in your page URLs.
> - Use a custom element for checking unique values of your URL slug elements. This gives content creators visual feedback to use another value if there's a conflict.
> - Check for the URL slug uniqueness in your app.

Preview URLs for the home page

The home page doesn't need a URL slug because it's static (it's unique and in one location). For example, a home page preview URL of an app running at `https://preview.example.com` would be `https://preview.example.com`.

Set up preview for Web Spotlight

When preparing embedded preview for your Web Spotlight project, you can set up preview links the same way as described in the previous section. But because the generated URLs will be loaded in an iframe within the Kontent app, you need to ensure a few things:

- Always use a secured connection (HTTPS) for your URL. For example, `https://preview.example.com`.
- Use the appropriate `frame-ancestors` directive in the `Content-Security-Policy` header to specify an allowed origin, in this case, `Content-Security-Policy: frame-ancestors https://app.kontent.ai`. For more details, see the MDN Web Docs.
- Make sure your web browser allows third-party cookies.
- If you first want to test your implementation locally, you need to generate a self-signed HTTPS certificate.

> Implement preview for Web Spotlight using Gatsby Starter.

Set up edit buttons in your preview

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

See the code example on https://codepen.io/KenticoKontent/pen/MWpXvQz

For content editors, this small addition means they can go straight to editing in Kontent 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 API endpoints.