Set up editing content directly from the preview

December 5, 2022 • Jan Cerman • 7 min read

Content creators sometimes need to correct errors or typos when they see them on preview of your website or app. Add edit buttons to your app to let your content creators quickly edit content in Kontent.ai.

And to make the authoring experience even better, set up automatic refresh for preview to see the latest changes without any hassle.

Automatically create edit links in web apps

Use the Smart Link SDK in the preview environment for your web apps to automatically create clickable regions on your pages.

Clicking the regions will open a specific content item in Kontent.ai. These regions will look like what you see in the Codepen example below.

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

These regions are based on annotations in your HTML code. For them to work, you need to annotate the HTML with data attributes so that the Smart link SDK knows where the content comes from.

Set up your web app for smart links

To create an edit link to a content item in Kontent.ai, the Smart link SDK needs to know several things such as your project ID, language, and content item ID. In your HTML code, you can also provide details about specific elements or components within a content item.

For example, you can annotate your HTML code like so:

1. In your website's `<body>`, specify the project ID using `data-kontent-project-id="1d50a0f7-9033-48f3-a96e-7771c73f9683"` and language using `data-kontent-language-codename="en-US"`.
2. In a `<div>`, specify the content item ID using `data-kontent-item-id="af858748-f48a-4169-9b35-b10c9d3984ef"`.
3. (Optional) In a `<p>`, specify the codename of an element using `data-kontent-element-codename="body"`.

When you initialize the Smart link SDK, it looks at these data attributes and uses them to create a URL such as `https://app.kontent.ai/goto/edit-item/project/1d50a0f7-9033-48f3-a96e-7771c73f9683/variant-codename/default/item/af858748-f48a-4169-9b35-b10c9d3984ef/element/body`.

You can also specify identifiers for content components using `data-kontent-component-id` so that your content creators can enjoy even better editing experience.

For more specific steps and examples, check out the Smart link SDK usage instructions.

Make sure you render the data attributes and edit links only in preview environment. In most cases, you want to display the edit links only in a preview environment that your editors can access. You can also use them in production – it's up to you to limit their visibility according to your needs.

Set up your web app for Add buttons

The ⊕ **Add** buttons in Web Spotlight can be set up in a similar way to edit links:

1. Provide your project ID, language, and a content item ID.
   - For details, see Set up your web app for smart links.
2. Specify the element, either a rich text or a linked items element, by using `data-kontent-element-codename`.
3. Use `data-kontent-add-button` to allow rendering of the **Add** button.
4. (Optional) Use `data-kontent-add-button-render-position` to specify the exact location of the **Add** button on preview.
5. Define where the new content should be added.
   - For a fixed position within the element, use `data-kontent-add-button-insert-position=start|end`.
   - For a relative position, use `data-kontent-add-button-insert-position=before|after` accompanied by `data-kontent-item-id` or `data-kontent-component-id` that specifies a linked item or a component before or after which you want to add new content.

For more specific steps and examples, check out the Smart link SDK usage instructions.

Manually create edit links in native apps

In native apps that don't return HTML, you need to construct the edit links manually. The edit links are based on a pattern with variables that you fill in based on what content your app displays on preview.

Create links to content items

To create edit links for specific content items, construct URLs with the following structure.

HTML

```
# Pattern
https://app.kontent.ai/goto/edit-item/project/<YOUR_PROJECT_ID>/variant-codename/<LANGUAGE_CODENAME>/item/<ITEM_ID>

# Example
https://app.kontent.ai/goto/edit-item/project/8d20758c-d74c-4f59-ae04-ee928c0816b7/variant-codename/en-US/item/4d4dc14d-8c7c-471f-b797-c6694c604964
```

Create links to elements in content items

To create edit links to specific elements within items, extend URL to a content item above with information about the specific element.

HTML

```
# Pattern
https://app.kontent.ai/goto/edit-item/project/<YOUR_PROJECT_ID>/variant-codename/<LANGUAGE_CODENAME>/item/<ITEM_ID>/element/<ELEMENT_CODENAME>

# Example
https://app.kontent.ai/goto/edit-item/project/8d20758c-d74c-4f59-ae04-ee928c0816b7/variant-codename/en-US/item/4d4dc14d-8c7c-471f-b797-c6694c604964/element/related_articles
```

When users click the edit links, Kontent.ai checks if the user has access to the specified content item and element.

Interactive example

See the code example on https://stackblitz.com/edit/typescript-kk-edit-button

Create links to nested content items

You can also link to a content item nested in a linked items or rich text element in another item. The structure of the linked relationship will be reflected in Kontent.ai in the breadcrumbs for easy navigation.

HTML

```
# Pattern
https://app.kontent.ai/goto/edit-item/project/<YOUR_PROJECT_ID>/variant-codename/<LANGUAGE_CODENAME>/item/<ITEM_ID>/element/<ELEMENT_CODENAME>/item/<NESTED_ITEM_ID>/element/<ELEMENT_CODENAME>

# Example
https://app.kontent.ai/goto/edit-item/project/8d20758c-d74c-4f59-ae04-ee928c0816b7/variant-codename/en-US/item/4d4dc14d-8c7c-471f-b797-c6694c604964/element/related_articles/item/29b82988-fb9e-4327-a34b-b568cfaa39e9/body
```

No edit links for content components

You can't link directly to elements nested in content components. You can only link to the highest-level rich text element in which the component is nested.

Automatically refresh preview in Web Spotlight

Automatic refresh in Web Spotlight makes live-editing easier by automatically updating the preview after each content change, without having to do this manually.

For your web apps to support automatic refresh for the preview, make sure that your preview environment:

1. Uses the latest version of the Smart Link SDK (autorefresh is supported from version 2.2.0).
2. Has the `X-KC-Wait-For-Loading-New-Content` header set to `true` when fetching data from the Delivery Preview API.

If both previously mentioned conditions are met, Web Spotlight will wait for your changes to be ready via the Delivery Preview API, and after that, the preview will be refreshed automatically.

Implement a custom refresh handler

In some cases, simply refreshing the page might not be enough. For example, if you use a static site generator for your preview, you need to trigger the rebuild of the page before refreshing it. Or maybe you don't want to refresh the entire page when a single item has been updated and would rather re-render only the affected place in the UI.

That's why the Smart Link SDK supports the custom refresh handler, which allows you to specify how your web page reacts to refresh events received from Web Spotlight.

TypeScript

```typescript
import KontentSmartLink, { KontentSmartLinkEvent } from '@kontent-ai/smart-link';

const sdk = KontentSmartLink.initialize({ ... });

sdk.on(
  KontentSmartLinkEvent.Refresh,
  (data, metadata, originalRefresh) => {
    // Custom refresh logic goes here
    // ...
    originalRefresh(); // If you want to refresh the page at the end
  }
);
```

> The ☑ **Changes are ready** status in Web Spotlight is displayed when the changes are available in the Delivery Preview API, not when the preview page has already been refreshed. If your custom refresh handler takes some time to finish, make sure to inform your users by displaying a loader or message inside the preview.

For more information about implementing your custom refresh handler, check out the Smart Link SDK usage instructions.

What's next?

- Try Web Spotlight for a seamless web editing experience.
- Run one of our sample applications to see the smart edit buttons in action.
- Set up preview for your content for your editors so that they review content with confidence.
- Secure access to your project to make sure only apps can get it.