Import content model

March 21, 2023 • Matus Usiak • 4 min read • JavaScript

Learn how to import content model using Management API. You'll go step by step and create a basic content type with several elements, a snippet, and a taxonomy. For the basics of using the Management API, first see Importing to Kontent.ai.

New to content modeling?

If you're starting out with content modeling, read more about what content modeling is and how to model your content right.

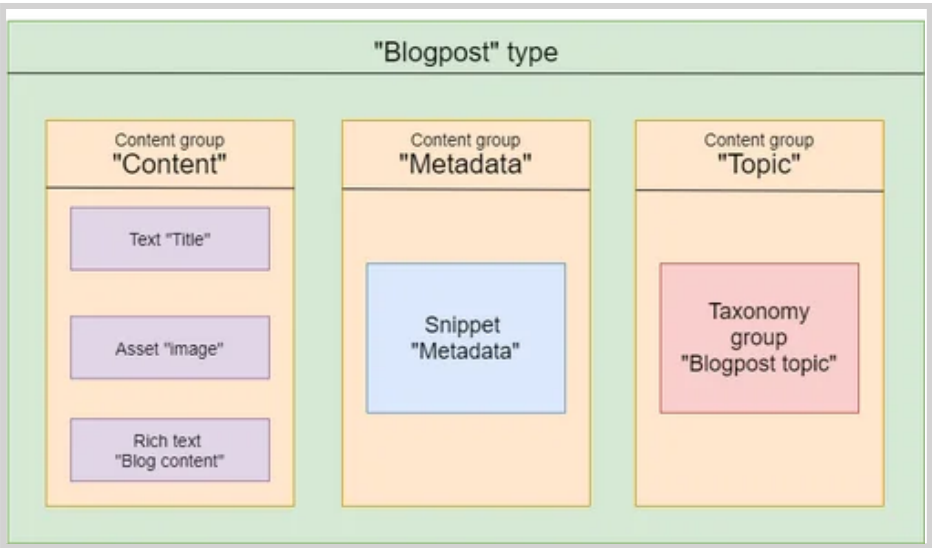
If you already know the basic principles and want to go further, take our content modeling e-learning series.

Content model overview

Before creating the content type itself, create the snippet and taxonomy group first. This way you can include them in the type as elements later on. When importing your content, it is better to start with these smaller parts (taxonomy group and snippet with elements) before you tie them all together.

When you think about re-using content, for example, using the same content elements in multiple items, it is easier to create the snippet with those elements and then include the snippet in all types the elements needs to be in. Rather than creating the types and later adding the elements to all of them.

In the diagram below, there's a *Blogpost* type with elements divided in three content groups. The first *Content* content group has some general fields such as *title* and *blog content*. In the *Metadata* content group, there's a snippet which can be re-used across multiple types. The third group named *Topic* contains a taxonomy, which can be used for tagging and organizing content.



1. Add a content type snippet

Let's start with adding the snippet called *Metadata* with two elements.

To add a snippet, make a POST request with the following properties:

- Name which serves as a display name of the snippet.
- An array of content elements you want to have in the snippet.

JavaScript

```
// Tip: Find more about JS/TS SDKs at https://kontent.ai/learn/javascript
// Using ES6 syntax
import { ManagementClient } from '@kontent-ai/management-sdk';

const client = new ManagementClient({
  environmentId: '<YOUR_ENVIRONMENT_ID>',
  apiKey: '<YOUR_API_KEY>'
});

const response = client.addContentTypeSnippet()
  .withData(builder => {
    return {
      name: "Metadata",
      codename: "metadata",
      elements: [
        builder.textElement({
          name: "Title",
          codename: "title",
          type: 'text'
        }),
        builder.textElement({
          name: "Keywords",
          codename: "keywords",
          type: 'text'
        }),
        builder.textElement({
          name: "Description",
          codename: "description",
          type: 'text'
        })
      ]
    };
  })
  .toPromise();
```

2. Add a taxonomy group

Now you can move onto adding a taxonomy group along with its terms. In this example, the taxonomy is called *blogpost topic* with terms like *sport* and *technology stack*. Both of these terms will serve as subgroups that will hold actual sports and programming languages. Taxonomies allow you to organize your content into hierarchies.

The sample taxonomy hierarchy looks like this:

- Blogpost topic
  - Sport
    - Soccer
    - Ice hockey
    - Rugby
  - Technology stack
    - JavaScript
    - C#
    - MVC

To add the taxonomy group with its terms, make a POST request with these properties:

- Name which serves as a display name of the taxonomy group.
- An array of taxonomy terms which can have more terms within them, like *JavaScript* under *Technology stack*.

JavaScript

```
// Tip: Find more about JS/TS SDKs at https://kontent.ai/learn/javascript
// Using ES6 syntax
import { ManagementClient } from '@kontent-ai/management-sdk';

const client = new ManagementClient({
  environmentId: '<YOUR_ENVIRONMENT_ID>',
  apiKey: '<YOUR_API_KEY>'
});

const response = client.addTaxonomy()
  .withData(
    {
      name: 'Blogpost topic',
      codename: 'blog_topic',
      terms: [
        {
          name: 'Sport',
          codename: 'sport',
          terms: [
            {
              name: 'Soccer',
```

```
            codename: 'soccer',
            terms: []
          },
          {
            name: 'Ice hockey',
            codename: 'hockey',
            terms: []
          },
          {
            name: 'Rugby',
            codename: 'rugby',
            terms: []
          }
        ]
      },
      {
        name: "Technology stack",
        codename: "tech",
        terms: [
          {
            name: "JavaScript",
            codename: "js",
            terms: []
          },
          {
            name: "C#",
            codename: "c",
            terms: []
          },
          {
            name: 'MVC',
            codename: 'mvc',
            terms: []
          }
        ]
      }
    ]
  }
)
.toPromise();
```

3. Add a content type

Finally, add the content type along with its content groups, as well as the content type snippet and taxonomy group you created earlier.

To add the content type with its elements, make a POST request with these properties:

- Name which serves as a display name of the content type.

- (Optional) The codename of the content type. If you don't specify the codename, it will be generated from the content type's display name. However, you can change the codename later.
- An array of content groups with these properties:
  - Name used as a display name.
  - External ID of the content group. You'll reference the external ID when adding content elements to specific content groups.
- An array of content elements that define structure of the type.

JavaScript

```
// Tip: Find more about JS/TS SDKs at https://kontent.ai/learn/javascript
// Using ES6 syntax
import { ManagementClient } from '@kontent-ai/management-sdk';

const client = new ManagementClient({
  environmentId: '<YOUR_ENVIRONMENT_ID>',
  apiKey: '<YOUR_API_KEY>'
});

const response = client.addContentType()
  .withData(builder => {
    return {
      name: 'Blogpost',
      codename: 'blogpost',
      content_groups: [{
        name: 'Content',
        external_id: 'content'
      },
      {
        name: 'Metadata',
        external_id: 'metadata'
      },
      {
        name: 'Topic',
        external_id: 'topic'
      }
      ],
      elements: [
        builder.textElement({
          name: 'Title',
          type: 'text',
          content_group: {
            external_id: 'content'
          },
        }),
        builder.assetElement({
          name: 'Image',
          type: 'asset',
```

```
          content_group: {
            external_id: 'content'
          },
        }),
        builder.richTextElement({
          name: 'Blog content',
          type: 'rich_text',
          content_group: {
            external_id: 'content'
          },
        }),
        builder.snippetElement({
          snippet: {
            'codename': 'metadata'
          },
          type: 'snippet',
          codename: 'metadata',
          content_group: {
            external_id: 'metadata'
          },
        }),
        builder.taxonomyElement({
          taxonomy_group: {
            'codename': 'blog_topic'
          },
          type: 'taxonomy',
          codename: 'taxonomy',
          content_group: {
            external_id: 'topic'
          }
        })
      ]
    };
  })
  .toPromise();
```

What's next?

- Update your content types programmatically and learn about the structure of content type elements.
- Take a look at further capabilities of the Management API when it comes to content types, snippets, and taxonomies.
- Check out the tools to use for migrating content from other systems to Kontent.ai or between Kontent.ai projects.
- If you've already got a content model of your own, use our content modeling checklist to make sure you follow best practices.