Set up sharing content between multiple projects

February 2, 2023 • Martina Farkasova • 6 min read • .NET

When managing multiple projects under one subscription, you might want to share some of the information from one project with another. Learn how you can leverage what Kontent.ai has to offer to make your projects cooperate smoothly.

Having a set of information that is shared across multiple projects has its advantages. For one, you can avoid replication of the data and save yourself the hassle of duplicating your content. Another benefit would be not adding more complexity when managing such information in separate projects.

For example, imagine you have multiple websites under your subscription and each of them utilizes the same data for localization strings. In this case, it would be easier to have this type of data in one shared project and pull that data whenever needed.

Decisions, decisions

It's always good to think about your particular use case before you start implementing a custom solution. When you have a set of content items from one project that you want to reuse in another project, you have a couple of options.

You might want to keep the related content items separate and only link them in your new project. In this case, using a custom element would be the preferred choice. This element lets you link published content items from a different project directly in the UI.

Still, sometimes it's better to have all the information in the same project even if it means duplicating content. This is especially true if you know you'll make changes to the related content and want to keep it at hand. Or if the project you're reusing data from is somehow restricted and you don't want content contributors to make changes to it. In situations like these, it might be helpful to clone an existing project that already contains the required data and then tweak the project to fit your needs.

If you have specific requirements for the format of your shared data or would like to pull content from multiple projects at once, it's best to do so programmatically. While taking advantage of the Kontent Delivery SDKs, you can retrieve content from multiple Delivery Clients.

> **Do you need multiple projects at all?**
>
> Use collections to separate content shared accross your organization from the department-specific content. Each department can then use its own collection so you can have one well organized project even for a massive corporate website or app.

As you can see, it all comes down to how you want to work with the content later. This article looks closely at two different options – one using a custom element in the UI and the other one using SDKs to pull content from multiple projects.

Linking content from a different project

If you need to link content items from a different project right in the UI, you can leverage a custom element designed specifically for this use case.

> **Ensure secure hosting**
>
> You need to host your custom element so the browser can fetch the file. The hosting of the element's source code is done on your end.

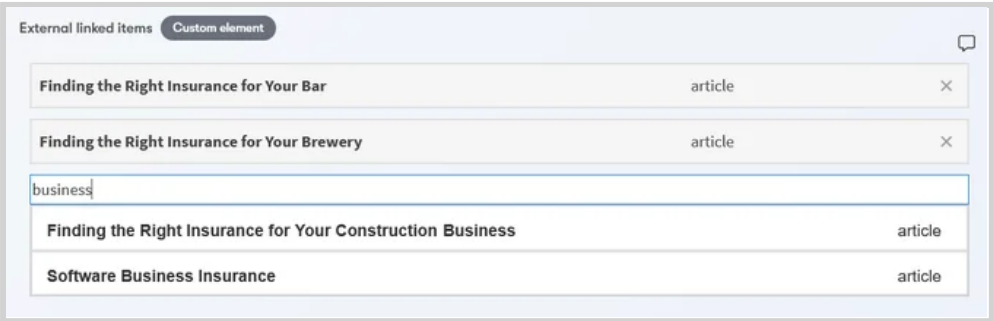
To add a custom element:

1. From the app menu, choose **Content model**.
2. Open the content type that should hold the custom element.
3. Drag in a *Custom element* from the right and give it a name.
4. Add your **hosted code URL**.
   - Make sure you have the source code hosted somewhere secure and get its URL.
5. Fill in the JSON **parameters**.
   - Specify the project containing the items you want to link by adding its `projectid`.
   - (Optional) Filter content items you want to retrieve using the `filter` parameter.
6. Click **Save changes**.

When using this custom element, there are a few things to keep in mind:

- You can only link published content items.
- If you plan on using the sample element in your own production project, we recommend you clone the repository. By doing so, you will not be affected by possible changes made to the custom element in the future.

Once configured, the custom element could look similar to this one:



For more details on how to work with custom elements, see Integrating your content editing features.

Getting content from multiple Delivery clients

Whether you're using the Delivery .NET SDK, Delivery JavaScript SDK, or another SDK, the principle of pulling the data from two projects stays the same. You first create two individual instances of `DeliveryClient`, supplying each with the corresponding project ID. Then you retrieve content items from both clients and get the response. In the code samples below, the response is an array containing items from both projects. But you can always change the format depending on how you want to work with the items.

If you don't want to get all the content items from the projects, you can always filter the results by, for example, specifying a content type.

C#

```
// Tip: Find more about .NET SDKs at https://kontent.ai/learn/net
using Kontent.Ai.Delivery;

public class Startup
{
    public IConfigurationRoot Configuration { get; }

    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    public void ConfigureServices(IServiceCollection services)
    {
        // Registers named clients based on the DeliveryOptions objects defined in appsettings.json
        // See https://kontent.ai/learn/net-register-multiple-clients for details
        services.AddDeliveryClient("first_project", Configuration, "DeliveryOptionsForFirstProject");
        services.AddDeliveryClient("second_project", Configuration, "DeliveryOptionsForSecondProject");
    }
}

public class YourController : Controller
{
    private IDeliveryClient client1;
    private IDeliveryClient client2;

    public YourController(IDeliveryClientFactory deliveryClientFactory)
    {
        // Creates instances of Delivery clients based on their names
        client1 = deliveryClientFactory.Get("first_project");
        client2 = deliveryClientFactory.Get("second_project");

```

```
        // Gets content items from both projects
        // Using the generic <object> produces strongly typed objects based on
"system.type"
        var response1 = await client1.GetItemsAsync<object>();
        var response2 = await client2.GetItemsAsync<object>();

        // Merges the responses
        IList<object> items = response1.Items.Concat(response2.Items).ToArray();
    }
}
```

Getting content from three or more different projects is fairly similar to getting content from two projects. You can apply the principles described above to suit your needs.

What's next?

In this tutorial, you've learned about the different approaches you can take when sharing content between projects.

- Read about getting content from one project and see how to filter and sort content items.
- Learn about mapping retrieved content to strongly typed models to streamline your development process.
- Save time when creating new projects by cloning existing projects.
- Implement your own custom elements, or see the integrations topic on GitHub and the custom elements overview to find custom elements you can use or customize for your scenario.