Sourcing from Kontent by Kentico

In this guide, you'll go through how to quickly source content for your Gatsby site from Kontent by Kentico.

Kontent is a hosted CMS that offers you Content as a Service (CaaS) for all your content needs. Using CaaS ensures that your content will be future-proof and reusable, so you can add a mobile app that uses the same content as your Gatsby site without worrying about how it will fit. Kontent offers CaaS with an easy-to-use editing interface and excellent collaboration features, so all your work on content can happen in one place without requiring technical help for each user.

You can also take advantage of Kontent's ability to deliver content in multiple languages and create relationships among various content with linked items. However you structure your content in Kontent, the source plugin will ensure the proper nodes are created for your Gatsby site.

Note: For this guide, you'll work from scratch from the Gatsby default starter to get a feel for how sourcing from Kontent works. If you'd like to look at an example of a complete basic site, have a look at the Kontent Gatsby starter site, which includes worked examples for querying content. Or you could also check out the Kontent starter based on Lumen template, which is a minimal, lightweight and mobile-first starter for creating blogs using Gatsby and Kontent.

Setup

Kontent

The first thing to do, if you haven't already done so, is sign up for a Kontent account. This will automatically start a free 30-day trial with all Kontent features. At any point during the trial or after, you can switch to a Starter plan (which always starts as free) or a higher plan with more features.

Once you have a subscription to Kontent, you need some content to retrieve. If you know what you want, you can set up your own content types (templates for your content) and then create content items (the actual content) based on them. If you'd like to take a shortcut and see some example content, you can use Sample Project generator to generate project "Sample Project" and import

sample content. This guide will continue with the example of the "Sample Project".

The created Sample Project is a comprehensive presentation of a fictional coffee company named Dancing Goat that showcases various Kontent features. It can be displayed in various channels, as you can see by going to the Quickstart page in Kontent from within that project.

For this guide, you don't have to worry about most of the features. You'll pull some data to display within the Gatsby site you'll create in the next step. The only thing you need to continue is your Project ID, which you can find in Kontent under *Project settings -> API keys*.

Gatsby

Adding content to existing pages Now that you have some content to pull, you can create a basic Gatsby site to display the content. Assuming you have the Gatsby CLI installed, create a new site and navigate to it in your terminal:

```
gatsby new kontent-guide
cd kontent-guide
```

Next, install the Kontent source plugin:

```
npm install @kentico/gatsby-source-kontent
```

Once that's done, you need to add the plugin to `gatsby-config.js`:

```
module.exports = {
  siteMetadata: {
    // ...
  },
  plugins: [
    // ...
    {
      resolve: `@kentico/gatsby-source-kontent`,
      options: {
        projectId: `<YourProjectID>`, // Fill in your Project ID
        // Please note that with the Sample Project generated above, `en-US` is the default
        languageCodenames: [
          `en-US`, // Or the languages in your project (Project settings -> Localization)
        ],
      },
    },
    // ...
  ],
}
```

And that's enough for you to be able to access content from Kontent in your site. You can see this by starting Gatsby in development mode:

```
gatsby develop
```

To see all the content that's available from Kontent, you can test out GraphQL queries in GraphiQL at `http://localhost:8000/___graphql`. The queries generated from Kontent will be prefixed with `kontentItem` (for single nodes) or `allKontentItem`.

To see how to put that data into your site, first go to `http://localhost:8000/`. Notice that the default title for the site is "Gatsby Default Starter". You can change that by pulling the title for your site from Kontent.

The title here is generated in the layout from the site metadata. By default, the Kontent Sample Project has a single item named "Home" that is the only item of the Home type. So you can change the layout component to query the metadata of that item and then use that data to populate your title.

```
// ...
const Layout = ({ children }) => {
  const data = useStaticQuery(graphql`
    query SiteTitleQuery{
      kontentItemHome {
        elements {
          metadata__meta_title {
            value
          }
        }
      }
    }
  `)

  return (
    <>
      <Header siteTitle={data.kontentItemHome.elements.metadata__meta_title.value} />
// ...
```

If you look at `http://localhost:8000/`, you'll notice the title is now "Dancing Goat–Freshest coffee on the block!". You can change this title in Kontent to whatever you want and rerun `gatsby develop` to rebuild the site (see below about automatic builds).

So you've seen how to add content to existing pages in Gatsby using Kontent. Next, you will start creating new pages of your own.

Creating new pages It's great to be able to add content to existing static pages, but one of the great benefits of using CaaS is being able to define pages in Kontent and having them generated automatically. To see how, you'll add pages based on content from the Sample Project in the Article type (feel free to explore how these are structured in Kontent).

Start by making use of the URL pattern in the Article type to generate slugs for your Article nodes:

```
exports.onCreateNode = ({ node, actions: { createNodeField } }) => {
  if (node.internal.type === `kontent_item_article`) {
    createNodeField({
      node,
      name: `slug`,
      value: node.elements.url_pattern.value,
    })
  }
}
```

Now that you have a pretty way to define the path for your pages, you can create the pages programmatically:

```
const path = require(`path`) // highlight-line

exports.onCreateNode = ({ node, actions: { createNodeField } }) => {
  if (node.internal.type === `kontent_item_article`) {
    createNodeField({
      node,
      name: `slug`,
      value: node.elements.url_pattern.value,
    })
  }
}

// highlight-start
exports.createPages = async ({ graphql, actions }) => {
  const { createPage } = actions

  // Query data from Kontent
  const result = await graphql(`
    {
      allKontentItemArticle {
        edges {
          node {
            fields {
              slug
            }
          }
        }
      }
    }
  `)
```

```
  // Create pages
  result.data.allKontentItemArticle.edges.forEach(({ node }) => {
    createPage({
      path: node.fields.slug,
      component: path.resolve(`src/templates/article.js`),
      context: {
        slug: node.fields.slug,
      },
    })
  })
}
// highlight-end
```

Now create a basic template to display each article with a title and the body that you pull with a GraphQL query:

```
import React from "react"
import { graphql } from "gatsby"

import Layout from "../components/layout"

const Article = ({ data }) => {
  const item = data.kontentItemArticle.elements

  return (
    <Layout>
      <h1>{item.title.value}</h1>
      <div dangerouslySetInnerHTML={{ __html: item.body_copy.value }} />
    </Layout>
  )
}

export default Article

export const query = graphql`
  query articleQuery($slug: String!) {
    kontentItemArticle(fields: { slug: { eq: $slug } }) {
      fields {
        slug
      }
      elements {
        body_copy {
          value
        }
        title {
          value
        }
```

```
      }
    }
  }
`
```

When you rerun `gatsby develop`, you'll be able to see each article as a page with content pulled from Kontent. To see a list of all pages, visit `http://localhost:8000/asdf` (or any other url that generates a 404).

The body copy for this article comes from a rich text element in Kontent. Links and inline linked items (e.g., embedded videos) are not resolved by default for rich text elements. If you want to resolve them, you can query the required data in structured form for resolution and create your own React components. You could use Rich text element component that is a part of the @kentico/gatsby-kontent-components package.

Since the Kontent source plugin is defining the GraphQL schema for data from Kontent, you could use this schema and extend it according to your needs. There are some examples of what you could do in your application.

Now you know how to create pages programmatically and pull their content from Kontent. To get the most out of your CaaS, you'll want to also make sure your site builds automatically whenever published content changes inside Kontent.

Continuous deployment

To keep your site static but always up to date with the latest content from Kontent, it helps to set up automatic deployment whenever your published content changes. Here, you can see how to set that up using Netlify, but the principle is similar using other services like Gatsby Cloud, or Travis CI, as with another site sourced from Kontent.

For automatic deployment from Netlify, first store your site's source code in a Git provider such as GitHub. Then log in to Netlify (e.g., through the same Git provider), create a new site from Git, and choose your site's source code. Netlify should automatically detect that you're using Gatsby and include the `gatsby build` command. If so, go ahead and deploy your site. Your site will now automatically build whenever you push changes to the source code in the Git repository.

Now that your site's up and running, you need to set up automatic builds when published content in Kontent changes. First, in Netlify create a new build hook with a name like "Change in Kontent content" and copy the URL. Then go to Kontent. Under *Project settings*, choose *Webhooks* and create a new webhook. Give it a name like "Netlify build", paste the URL into the *URL address* field and choose the events to trigger the webhook, you want to select just "DELIVERY API TRIGGERS" for content item events: "Publish" and "Unpublish". And that's it. Now whenever published content changes, your webhook will trigger a build in Netlify to ensure your static content is updated to the latest version.

What's next?

You've seen how to set up a Gatsby site that sources content from Kontent and is automatically redeployed on any change to the content. Kontent is capable of creating many other kinds of relationships, including taxonomies for categorization, multiple languages, and linking items together. Want to do more?

- See more about what the Kontent source plugin can do.
- Read the Kontent documentation to see what's possible.
- Explore the Kontent Gatsby starter to see a sample site.
- Explore the Kontent Gatsby starter Lumen to see a more complete example.