Sourcing from ButterCMS

Overview

In this guide you'll be setting up a CMS powered Gatsby site that uses ButterCMS as its content management system.

To complete this tutorial, you'll need your own ButterCMS auth token which you can get free here.

ButterCMS is a headless CMS that lets you manage content using their dashboard and integrate it into your tech stack of choice with their content APIs. You can use ButterCMS for new projects as well as add it to existing codebases.

ButterCMS provides a user-friendly UI for managing marketing sites, blogging, and custom content scenarios. It can be used for SEO landing pages, customer case studies, company news & updates, events + webinar pages, education center, location pages, knowledgebases, and more.

ButterCMS is different from a traditional CMS like Drupal or WordPress in that they're not a large piece of software you need to download, host, customize, and maintain. Instead, they provide consumable, performant content API's that you add to your application.

For example, if you wanted to enable a non-technical person to be able to add customer case study pages to your marketing site, you might create a Case Study Page Type to represent these pages. The non-technical person would be able to manage these pages from their dashboard and the JSON API output would look something like this:

```
{
  "data": {
    "slug": "acme-co-case-study",
    "fields": {
      "seo_title": "Acme Co Customer Case Study",
      "seo_description": "Acme Co saved 200% on Anvil costs with ButterCMS",
      "title": "Acme Co loves ButterCMS",
      "body": "<p>We've been able to make anvils faster than ever before! - Chief Anvil Make
    }
  }
}
```

Setup

Create a new Gatsby site with the default starter

Run this in your terminal:

```
gatsby new butter-site
```

Install the source plugin

```
npm install gatsby-source-buttercms
```

Adding configuration

Here you'll specify the config that will be needed to pull down data from ButterCMS. Make sure to add your **API_TOKEN** from your dashboard. In this guide you will be creating `faq_items`, `faq_headline`, `homepage`, `customer_case_study` as stated in the config below. Do well to change it if you named it something differently.

```
module.exports = {
{
      resolve: `gatsby-source-buttercms`,
      options: {
        authToken: `your_auth_token`,
        // Optional. Returns values for the supplied content field keys.
        contentFields: {
          keys: [`faq_items`, `faq_headline`],
          // Optional. Set to 1 to enable test mode for viewing draft content.
          test: 0,
        },
        // Optional. Array of page slugs.
        pages: [`homepage`],
        // Optional. Array of page types.
        pageTypes: [`customer_case_study`],
      },
    },
}
```

More details here

ButterCMS starter template

To see a fully complete Gatsby+ButterCMS project check out this Gatsby ButterCMS Starter Project. It contains real world examples of how to use Pages, Posts, and ContentFields.

Usage

Webhooks

Webhooks are a powerful feature that allow you to notify your internal systems whenever content in ButterCMS has changed. Your host platform needs to be notified so that Gatsby can create fresh pages from the new data. You can learn more about Webhooks in this blog post. Checkout your host platform from incoming webhooks so you can hit it anytime your content changes. Netlify lets you generate a build hook that will be triggered by ButterCMS on certain events e.g. when you create or update a blog post, more details here.

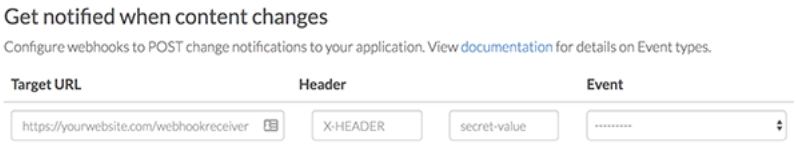


Figure 1: Webhook

Image transformation

ButterCMS has integrated with a rich image transformation API called Filestack. This allows you to modify your uploaded images in dozens of ways. Everything from resizing, cropping, effects, filters, applying watermarks and more. Check out Filestack full documentation for more detail.

After you upload an image to ButterCMS, it's stored on your CDN. To create a thumbnail, here's an example:

Original URL = https://cdn.buttercms.com/zjypya5tRny63LqhHQrv

Thumbnail URL = https://fs.buttercms.com/resize=width:200,height:200/zjypya5tRny63LqhHQrv

Resizing is just one of the many different transformations you can do to your images. Refer to the Filestack docs for full details.

Localization

ButterCMS has full support for localization of your content. Locale names and keys are completely customizable and there's no limit to the number of locales you can have. View their API Reference to learn how to query by locale.

Creating pages

Creating a single page (home page)

Introduction Quickly launch a new marketing site or add CMS-powered pages to your existing site using Pages.

Figure 2: locales

Adding a CMS-powered page to your app involves three steps:

1. Create the Page structure
2. Populate the content
3. Integrate into your application

Create the page structure Create a new Page and define its structure using Page Builder. Create an example homepage to follow along with this guide:

Populate the content Then populate your new page with content. In the next step, you'll call the ButterCMS API to retrieve this content from your app.

Integrate into your application With your homepage defined, the ButterCMS GraphQL query will return some data that looks like this:


```
{
  "data": {
    "slug": "homepage",
    "fields": {
      "seo_title": "Anvils and Dynamite | Acme Co",
      "headline": "Acme Co provides supplies to your favorite cartoon heroes.",
      "hero_image": "https://cdn.buttercms.com/c8oSTGcwQDC5I58km5WV",
      "call_to_action": "Buy Now",
      "customer_logos": [
        {
          "logo_image": "https://cdn.buttercms.com/c8oSTGcwQDC5I58km5WV"
```

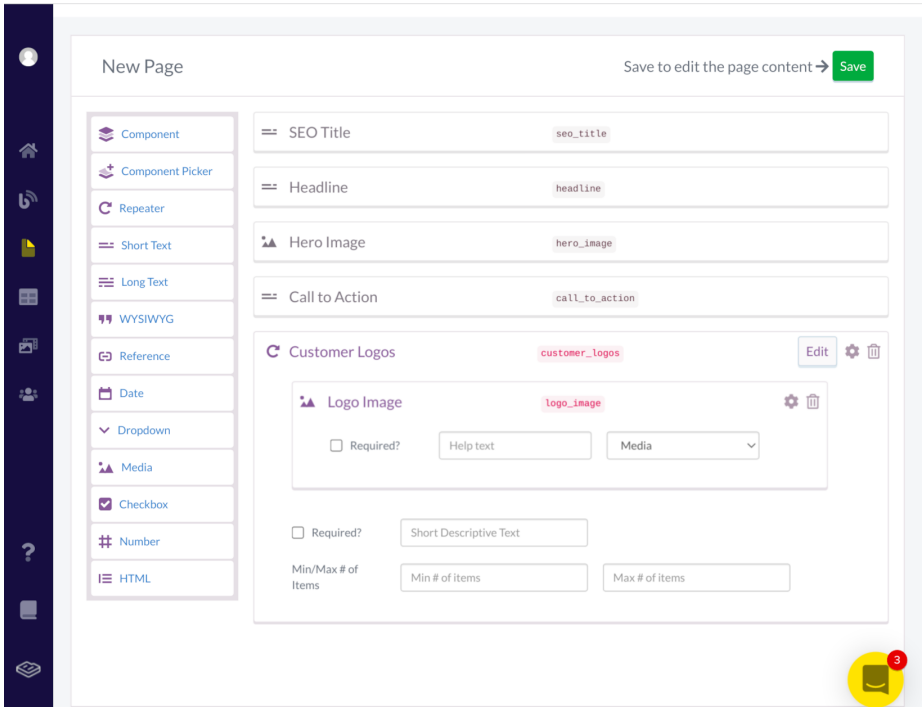


Figure 3: New page dashboard

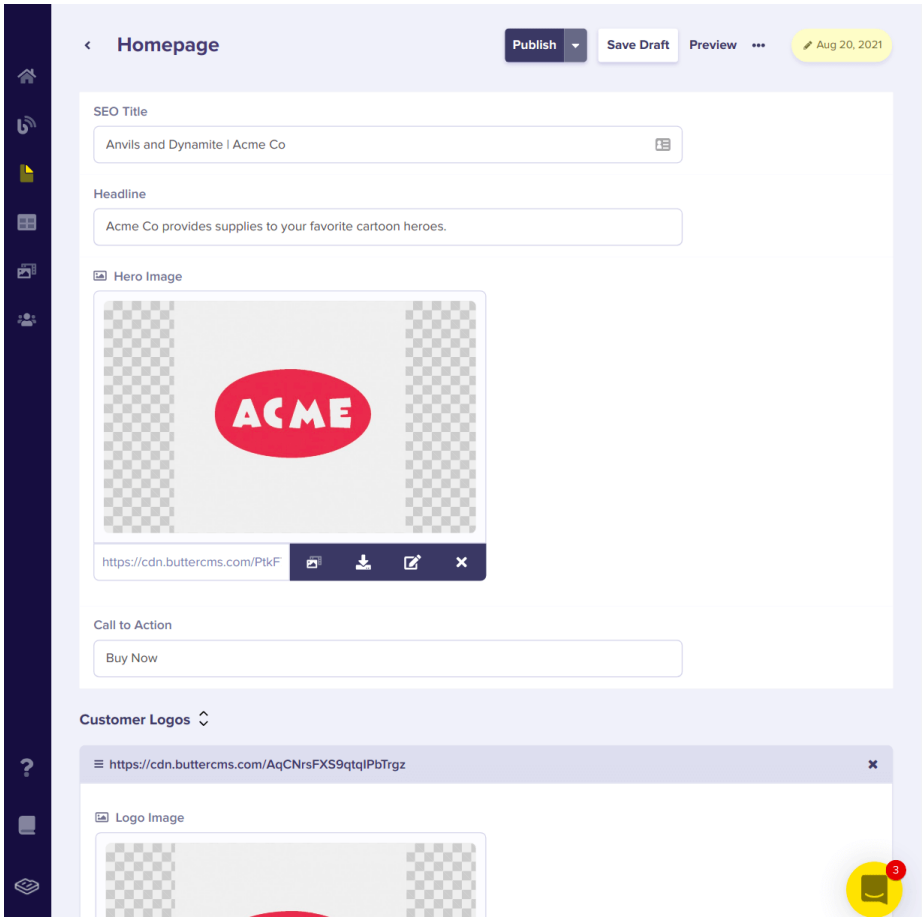


Figure 4: image

```
        },
        {
          "logo_image": "https://cdn.buttercms.com/c8oSTGcwQDC5I58km5WV"
        }
      ]
    }
  }
}
```

Now let's create the home page:

```
import React from "react"
import { graphql, Link } from "gatsby"
import Layout from "../components/layout"
import SEO from "../components/seo"

const IndexPage = ({ data }) => {
  const home = data.home.edges[0].node

  return (
    <Layout>
      <SEO
        title={home.seo_title}
        keywords={[`gatsby`, `application`, `react`]}
      />
      <div
        style={{
          height: `50%`,
          display: `flex`,
          padding: `1rem`,
          alignItems: `center`,
          justifyContent: `center`,
          flexDirection: `column`,
          background: `linear-gradient(-45deg, rgb(29, 64, 86) 0%, rgb(60, 24, 78) 100%)`,
        }}
      >
        <h1
          style={{
            textAlign: `center`,
            color: `white`,
            fontSize: `2.5rem`,
            fontWeight: `100`,
            maxWidth: `960px`,
          }}
        >
          {home.headline}
        </h1>
```

```
          <button
            style={{
              padding: `0.75rem`,
              backgroundColor: `white`,
              border: `none`,
              fontSize: `1.5rem`,
              borderRadius: `10px`,
            }}
          >
            {home.call_to_action}
          </button>
        </div>

        {/* <h1> {page.hero_image}</h1> */}

        <h1 style={{ fontWeight: `100`, textAlign: `center` }}>Our Customers</h1>
        <div
          style={{
            display: `flex`,
            flexDirection: `column`,
            alignItems: `center`,
            justifyContent: `center`,
          }}
        >
          {home.customer_logos.map(({ logo_image }) => (
            <img
              key={logo_image}
              style={{ width: `200px`, borderRadius: `10px` }}
              src={logo_image}
            />
          ))}
        </div>
      </Layout>
    )
}

//GraphQl query to fetch homepage data
export const query = graphql`
  {
    home: allButterPage(filter: { slug: { eq: "homepage" } }) {
      edges {
        node {
          slug
          headline
          seo_title
          customer_logos {
```

```
            logo_image
          }
          hero_image
          call_to_action
        }
      }
    }
  }
`

export default IndexPage
```

in your terminal, run

```
gatsby develop
```

Now open up `http://localhost:8000/home` to see the home page populated with the content you created on butter.

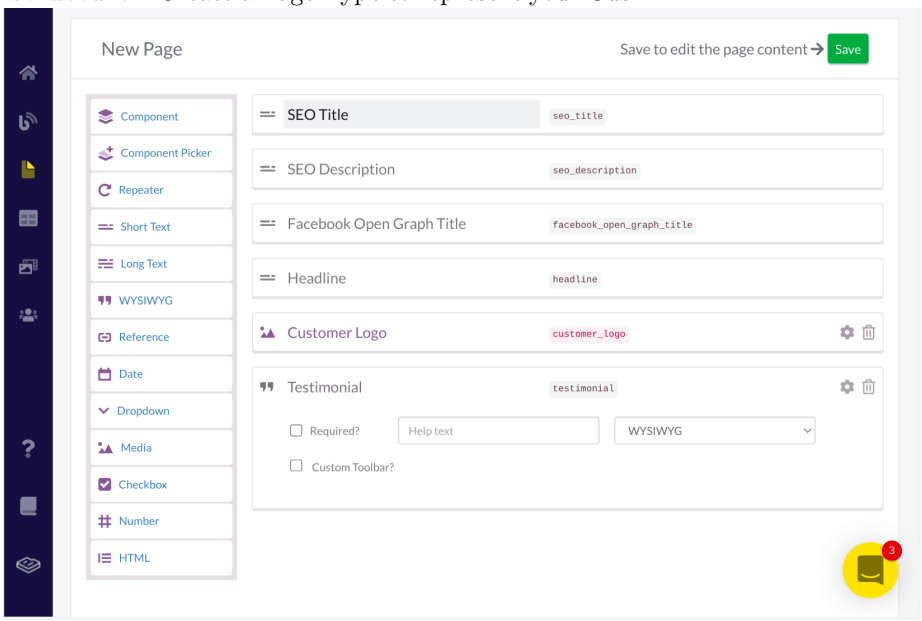
Create multiple pages using Page Types

Suppose you want to add a set of customer case study pages to your marketing site. They all have the same structure but the content is different. Page Types are perfect for this scenario and involves three steps:

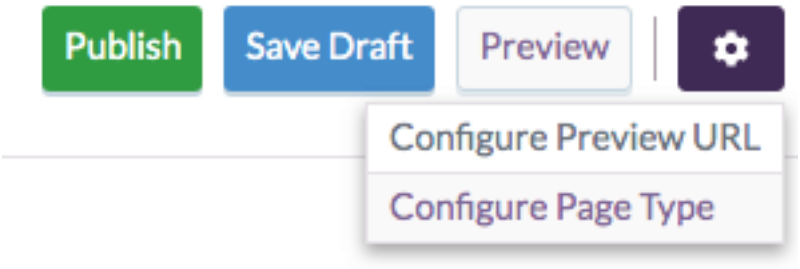
1. Create the Page Type structure
2. Populate the content
3. Integrate into your application

If you need help after reading this, contact ButterCMS via email or livechat.

Create the Page Type structure Create a Page Type to represent your Customer Case Study pages:



After saving, return to the configuration page by clicking the gear icon:



Then click on Create Page Type and name it "Customer Case Study". This will allow you to reuse this field configuration across multiple customer case study pages:

Populate the content Then populate the new page with content. In the next step, you'll call the ButterCMS API to retrieve this content from your app.

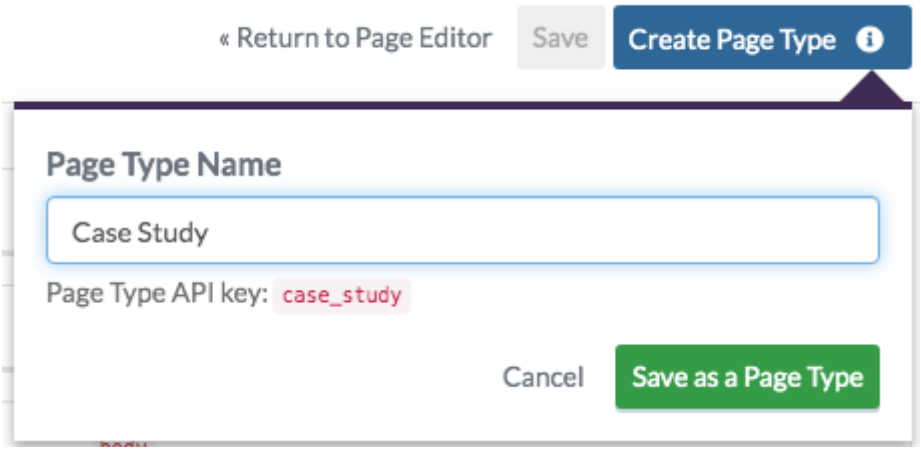
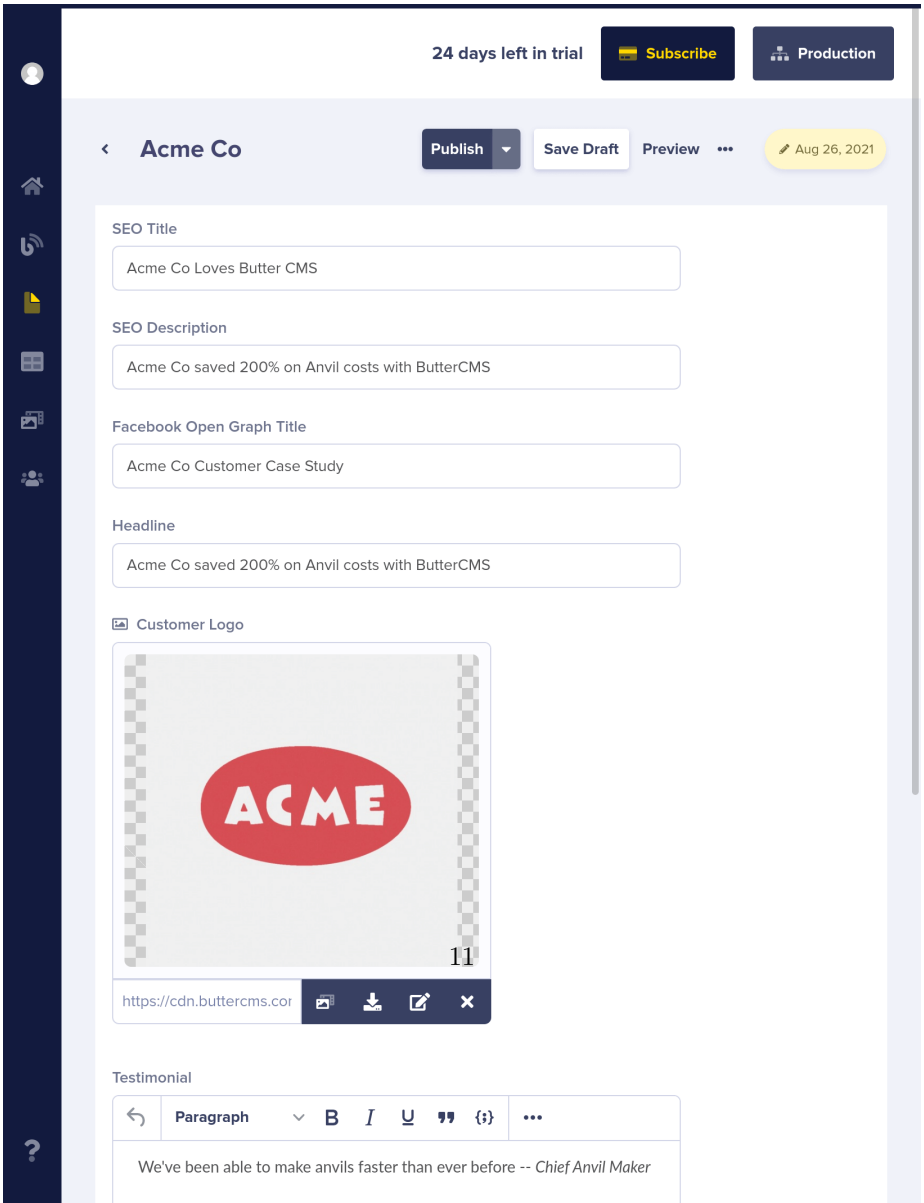


Figure 5: saving



To pull down content into Gatsby, run:

```
gatsby develop
```

Testing with GraphiQL

You can test out your GraphQL queries with GraphiQL (a GraphQL debugger) fire up GraphiQL on `http://localhost:8000/___graphql`

Once GraphiQL is open, paste the query below :

```
{
  allButterPage(filter: { page_type: { eq: "customer_case_study" } }) {
    edges {
      node {
        id
        facebook_open_graph_title
        seo_title
        headline
        customer_logo
        testimonial
      }
    }
  }
}
```

Integrate into your application Now refactor the home page to display link(s) to each customer case study page

```
import React from "react"
import { graphql, Link } from "gatsby"
import Layout from "../components/layout"
import SEO from "../components/seo"

const IndexPage = ({ data }) => {
  console.log(data)
  const home = data.home.edges[0].node
  const case_studies = data.case_studies.edges

  return (
    <Layout>
      <SEO
        title={home.seo_title}
        keywords={[`gatsby`, `application`, `react`]}
      />
      <div
        style={{
          height: `50%`,
```

```
    display: `flex`,
    padding: `1rem`,
    alignItems: `center`,
    justifyContent: `center`,
    flexDirection: `column`,
    background: `linear-gradient(-45deg, rgb(29, 64, 86) 0%, rgb(60, 24, 78) 100%)`,
  }}
>
  <h1
    style={{
      textAlign: `center`,
      color: `white`,
      fontSize: `2.5rem`,
      fontWeight: `100`,
      maxWidth: `960px`,
    }}
  >
    {home.headline}
  </h1>
  <button
    style={{
      padding: `0.75rem`,
      backgroundColor: `white`,
      border: `none`,
      fontSize: `1.5rem`,
      borderRadius: `10px`,
    }}
  >
    {home.call_to_action}
  </button>
</div>

<h1 style={{ fontWeight: `100`, textAlign: `center` }}>Our Customers</h1>
<div
  style={{
    display: `flex`,
    flexDirection: `column`,
    alignItems: `center`,
    justifyContent: `center`,
  }}
>
  {home.customer_logos.map(({ logo_image }) => (
    <img
      key={logo_image}
      style={{ width: `200px`, borderRadius: `10px` }}
      src={logo_image}
```

```
          />
        ))}

        <h1 style={{ fontWeight: `100` }}>Case Studies</h1>
        {case_studies.map(({ node: { id, slug, headline } }) => (
          <div key={id}>
            <Link to={`case-study/${slug}`}>{headline}</Link>
          </div>
        ))}
      </div>
    </Layout>
  )
}

export const query = graphql`
  {
    home: allButterPage(filter: { slug: { eq: "homepage" } }) {
      edges {
        node {
          slug
          headline
          seo_title
          customer_logos {
            logo_image
          }
          hero_image
          call_to_action
        }
      }
    }
    case_studies: allButterPage(
      filter: { page_type: { eq: "customer_case_study" } }
    ) {
      edges {
        node {
          id
          slug
          facebook_open_graph_title
          seo_title
          headline
          testimony
          customer_logo
        }
      }
    }
  }
```

```
`

export default IndexPage
```

Next you'll refactor `gatsby-node-js` to programmatically create customer case study pages with Gatsby create pages API. First you need to define a customer case study template

```
import React from "react"
import { graphql } from "gatsby"
import Layout from "../components/layout"
import SEO from "../components/seo"

function CustomerCaseStudy({ data }) {
  const page = data.allButterPage.edges[0].node

  return (
    <Layout>
      <SEO title={page.facebook_open_graph_title} description={page.headline} />
      <div>
        <h1>{page.seo_title}</h1>
        <p>{page.headline}</p>
        <img alt="customer_logo" src={page.customer_logo} />
        <p>{page.testimonial}</p>
      </div>
    </Layout>
  )
}

export const pageQuery = graphql`
  query CaseStudyPageBySlug($slug: String!) {
    allButterPage(filter: { slug: { eq: $slug } }) {
      edges {
        node {
          id
          slug
          facebook_open_graph_title
          seo_title
          headline
          testimony
          customer_logo
        }
      }
    }
  }
`
```

```
export default CustomerCaseStudy
```

Now programmatically create customer case study pages based on the template you defined in `src/template/customer-case-study.js`

```
const path = require(`path`)

exports.createPages = async ({ graphql, actions }) => {
  const { createPage } = actions

  // Blog post template
  const blogPost = path.resolve(`./src/templates/blog-post.js`)

  //customer case study template
  const customerCaseStudy = path.resolve(
    `./src/templates/customer-case-study.js`
  )

  let posts
  try {
    posts = await graphql(`
      {
        allButterPost {
          edges {
            node {
              id
              seo_title
              slug
              categories {
                name
                slug
              }
              author {
                first_name
                last_name
                email
                slug
                profile_image
              }
              body
            }
          }
        }
      }
    `)
  } catch (error) {
    console.log(`Error Running Querying Posts`, error)
```

```
}

posts = posts.data.allButterPost.edges

posts.forEach((post, index) => {
  const previous = index === posts.length - 1 ? null : posts[index + 1].node
  const next = index === 0 ? null : posts[index - 1].node

  // Create blog posts pages.
  createPage({
    path: `/blog/${post.node.slug}`,
    component: blogPost,
    context: {
      slug: post.node.slug,
      previous,
      next,
    },
  })
})

// Fetch Customer Case study pages
let pages
try {
  pages = await graphql(`
    {
      allButterPage(filter: { page_type: { eq: "customer_case_study" } }) {
        edges {
          node {
            id
            slug
            facebook_open_graph_title
            seo_title
            headline
            testimony
            customer_logo
          }
        }
      }
    }
  `)
} catch (error) {
  console.log(`Error Running Querying Pages`, error)
}

//Create Customer Case study pages
pages.data.allButterPage.edges.forEach(page => {
```

```
    createPage({
      path: `/case-study/${page.node.slug}`,
      component: customerCaseStudy,
      context: {
        slug: page.node.slug,
      },
    })
  })
}
```

That's it! Now stop the server and run:

```
gatsby develop
```

Now the home page should contain links to customer case study pages, click around and you'll notice that the template you defined in `src/template/customer_case_study.js` was used by Gatsby to create each case study page.

FAQ page example

Suppose you want to add a CMS to a static FAQ page with a title and a list of questions with answers. Most websites have a FAQ (Frequently Asked Questions) page. ButterCMS makes it possible to create such content with Collections. Now you'll create a collection named `FAQs`having a `question` and `answer` field.

Set up content fields Making your content dynamic with Butter is a two-step process:

1. **Setup custom content fields in Butter**
2. **Integrate the fields into your application**
3. **To set up custom content fields, first sign in to the Butter dashboard.**

Create a new workspace or click on an existing one. Workspaces let you organize content fields in a friendly way for content editors and have no effect on development or the API. For example, a real-estate website might have a workspace called "Properties" and another called "About Page".

Once you're in a workspace click the button to create a new content field. Choose the "Object" type and name the field "FAQ Headline":

After saving, add another field but this time choose the "Collection" type and name the field FAQ Items:

On the next screen setup two properties for items in the collection:

Now go back to your workspace and update your heading and FAQ items.

workspace content

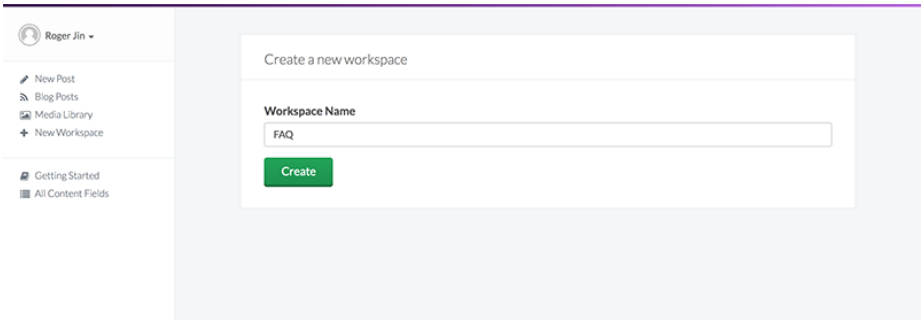


Figure 6: create workspace

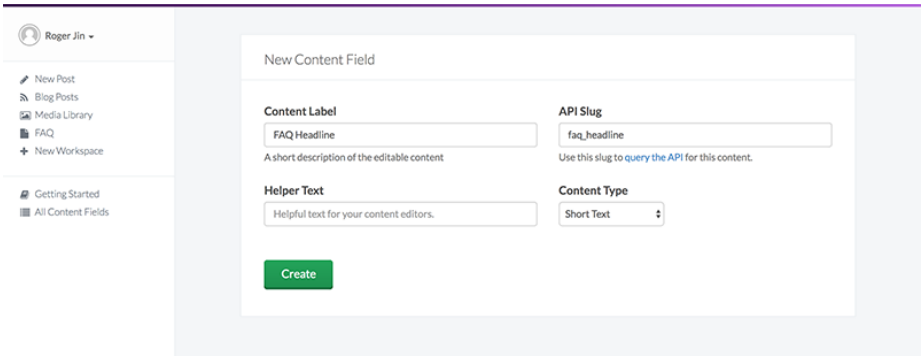


Figure 7: new contentfield

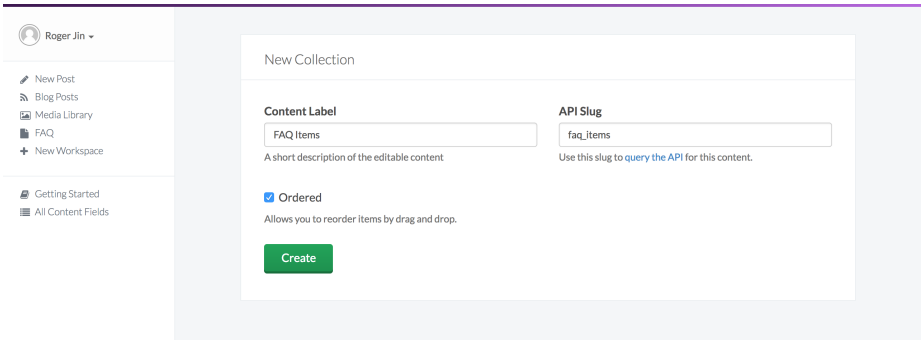


Figure 8: Add items

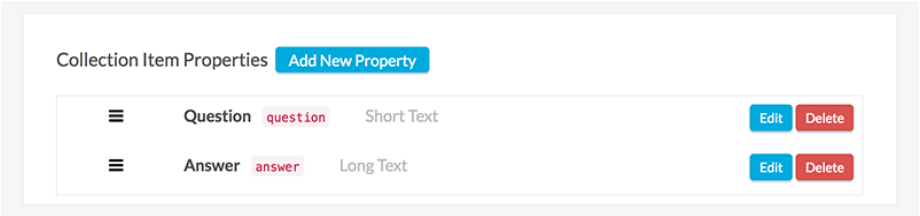


Figure 9: properties

Integrate into your application

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

import Layout from "../components/layout"
import SEO from "../components/seo"

const Faq = ({ data }) => {
  const FAQs = data.allButterCollection.edges[0].node.value
  const headline = data.allButterContentField.edges[0].node.value

  return (
    <Layout>
      <SEO title="FAQ - Frequently Asked Questions" />
      <h1
        style={{
          height: `30%`,
          color: `white`,
          display: `flex`,
          padding: `1rem`,
          alignItems: `center`,
          justifyContent: `center`,
          flexDirection: `column`,
          background: `linear-gradient(-45deg, rgb(29, 64, 86) 0%, rgb(60, 24, 78) 100%)`,
        }}
      >
        {headline}
      </h1>
      <div style={{ display: `flex`, padding: `10px` }}>
        {FAQs.map(faq => (
          <div
            style={{
              flexBasis: `50%`,
              padding: `10px`,
              background: `whitesmoke`,
```

```
              borderRadius: `10px`,
              margin: `5px`,
            }}
          >
            <h2 style={{ color: `#213b55` }}>{faq.question}</h2>
            <p style={{ fontSize: `1.5rem` }}>{faq.answer} </p>
          </div>
        ))}
      </div>
    </Layout>
  )
}

export const query = graphql`
  {
    allButterCollection(filter: { id: { eq: "faq_items" } }) {
      edges {
        node {
          id
          value {
            question
            answer
          }
        }
      }
    }

    allButterContentField(filter: { id: { eq: "faq_headline" } }) {
      edges {
        node {
          id
          value
        }
      }
    }
  }
`
export default Faq
```

Blog

Introduction

ButterCMS is also a great feat if you want to spin up a blog, which you can do through their provided blog engine that helps you manage content in one place. Gatsby then pulls down the data at build time and create static pages off that

data.

Blog home page

Now you will create a home page for your blog posts. It basically lists all blog posts.

```
import React from "react"
import { Link, graphql } from "gatsby"
import Layout from "../components/Layout"
import SEO from "../components/seo"

class BlogIndex extends React.Component {
  render() {
    const { data } = this.props
    const siteTitle = data.site.siteMetadata.title
    const posts = data.allButterPost.edges

    return (
      <Layout location={this.props.location} title={siteTitle}>
        <SEO
          title="All posts"
          keywords={[`blog`, `gatsby`, `javascript`, `react`]}
        />

        <div
          style={{
            alignItems: `center`,
            justifyContent: `center`,
            margin: `20px 0px 20px 0px`,
          }}
        >
          <div
            style={{
              maxWidth: `960px`,
              padding: `30px`,
            }}
          >
            {posts.map(({ node }) => {
              const title = node.seo_title || node.slug
              return (
                <div
                  key={node.slug}
                  style={{ margin: `10px`, padding: `10px` }}
                >
                  <h3>
```

```
                    <Link
                      style={{ boxShadow: `none` }}
                      to={`/blog/${node.slug}`}
                    >
                      {title}
                    </Link>
                  </h3>
                  <small>{node.date}</small>
                  <div
                    dangerouslySetInnerHTML={{ __html: node.meta_description }}
                  />
                </div>
              )
            })}
          </div>
        </div>
      </Layout>
    )
  }
}

export default BlogIndex

export const pageQuery = graphql`
  query {
    site {
      siteMetadata {
        title
      }
    }
    allButterPost {
      edges {
        node {
          id
          seo_title
          meta_description
          slug
          categories {
            name
            slug
          }
          author {
            first_name
            last_name
            email
            slug
```

```
            bio
            title
            linkedin_url
            facebook_url
            instagram_url
            pinterest_url
            twitter_handle
            profile_image
          }
          body
        }
      }
    }
  }
`
```

Creating a blog template

Now you've listed your blog posts in `src/pages/blog.js`, using gatsby createpages API you would generate blog post pages using a template:

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

import Bio from "../components/Bio"
import Layout from "../components/Layout"
import SEO from "../components/seo"

class BlogPostTemplate extends React.Component {
  render() {
    const post = this.props.data.allButterPost.edges[0].node
    const siteTitle = this.props.data.site.siteMetadata.title
    const { previous, next } = this.props.pageContext

    return (
      <Layout location={this.props.location} title={siteTitle}>
        <SEO title={post.seo_title} description={post.description} />
        <div
          style={{
            display: `flex`,
            alignItems: `center`,
            justifyContent: `center`,
            margin: `20px 0px 20px 0px`,
          }}
        >
          <div style={{ maxWidth: `960px`, padding: `30px` }}>
```

```
            <h1>{post.seo_title}</h1> <span>{post.date}</span> &bull;
            {post.categories.map(category => (
              <span>{category.name}</span>
            ))}
            <hr />
            <div
              style={{ paddingTop: `20px` }}
              dangerouslySetInnerHTML={{ __html: post.body }}
            />
            <hr />
            <Bio />
            <ul
              style={{
                display: `flex`,
                flexWrap: `wrap`,
                justifyContent: `space-between`,
                listStyle: `none`,
                padding: 0,
              }}
            >
              <li>
                {previous && (
                  <Link to={`/blog/${previous.slug}`} rel="prev">
                    ← {previous.seo_title}
                  </Link>
                )}
              </li>
              <li>
                {next && (
                  <Link to={`/blog/${next.slug}`} rel="next">
                    {next.seo_title} →
                  </Link>
                )}
              </li>
            </ul>
          </div>
        </div>
      </Layout>
    )
  }
}

export default BlogPostTemplate

export const pageQuery = graphql`
  query BlogPostBySlug($slug: String!) {
```

```
    site {
      siteMetadata {
        title
        author
      }
    }
    allButterPost(filter: { slug: { eq: $slug } }) {
      edges {
        node {
          id
          body
          seo_title
          date
          categories {
            name
          }
        }
      }
    }
  }
`
```

Generate blog pages

Now you'll use the blog template defined in `src/templates/blog-post.js` to generate blog pages.

```
const path = require(`path`)

exports.createPages = async ({ graphql, actions }) => {
  const { createPage } = actions

  const blogPost = path.resolve(`./src/templates/blog-post.js`)

  let posts
  try {
    posts = await graphql(`
      {
        allButterPost {
          edges {
            node {
              id
              seo_title
              slug
              categories {
                name
```

```
              slug
            }
            author {
              first_name
              last_name
              email
              slug
              profile_image
            }
            body
          }
        }
      }
    }
  `)
  } catch (error) {
    console.log(`Error Running Querying Posts`, error)
  }

  posts = posts.data.allButterPost.edges;

  posts.forEach((post, index) => {
    const previous = index === posts.length - 1 ? null : posts[index + 1].node
    const next = index === 0 ? null : posts[index - 1].node
}
```

Categories, tags, and authors

Use Butter's APIs for categories, tags, and authors to feature and filter content on your blog. See their API reference for more information about these objects:

Easy as Butter

This was an example meant to help you understand how ButterCMS works with Gatsby. You're now able to:

- Create a ButterCMS repository and set it up together with the Gatsby plugin
- Query data from ButterCMS for single pages, multiple pages, blog posts, and custom content fields

If you got stuck, you can compare your code to the gatsby-starter-buttercms. To learn more about ButterCMS, check out their blog. Their latest updates can be found on buttercms.com.