Gatsby and its ecosystem of plugins provide all kinds of data through a GraphQL interface. This guide will show how that data can be used to programmatically create pages.

> ***Note:*** *For most use cases you'll be able to use the File System Route API to create pages. Please read on if you need more control over the page creation or consume data outside of Gatsby's GraphQL data layer.*

Prerequisites

Though you can use any data source you'd like, this guide will show how to create pages from Markdown files (following after the example introduced in earlier guides).

Creating pages

The Gatsby Node API provides the `createPages` extension point which you'll use to add pages. This function will give you access to the `createPage` action which is at the core of programmatically creating a page.

```
exports.createPages = async function ({ actions, graphql }) {
  const { data } = await graphql(`
    query {
      allMarkdownRemark {
        edges {
          node {
            fields {
              slug
            }
          }
        }
      }
    }
  `)
  // highlight-start
  data.allMarkdownRemark.edges.forEach(edge => {
    const slug = edge.node.fields.slug
    actions.createPage({
      path: slug,
      component: require.resolve(`./src/templates/blog-post.js`),
      context: { slug: slug },
    })
  })
  // highlight-end
}
```

For each page you want to create you must specify the `path` for visiting that page, the `component` template used to render that page, and any `context` you need in the component for rendering.

The `context` parameter is *optional*, though often times it will include a unique identifier that can be used to query for associated data that will be rendered to the page. All `context` values are made available to a template's GraphQL queries as arguments prefaced with `$`, so from our example above the `slug` property will become the `$slug` argument in our page query:

```
export const query = graphql`
  query($slug: String!) {
    ...
  }
`
```

Specifying a template

The `createPage` action requires that you specify the `component` template that will be used to render the page. Here is an example of what the referenced template could look like:

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

export default function BlogPost({ data }) {
  const post = data.markdownRemark
  return (
    <Layout>
      <div>
        <h1>{post.frontmatter.title}</h1>
        <div dangerouslySetInnerHTML={{ __html: post.html }} />
      </div>
    </Layout>
  )
}

export const query = graphql`
  query($slug: String!) {
    markdownRemark(fields: { slug: { eq: $slug } }) {
      html
      frontmatter {
        title
      }
    }
  }
`
```

Notice that you're able to query with the `$slug` value from your `context` as an argument, which ensures that you're returning only the data that matches that specific page. As a result, you can provide the `title` and `html` from the matching `markdownRemark` record to your component. The `context` values are also available as the `pageContext` prop in the template component itself.

Not just Markdown

The `gatsby-transformer-remark` plugin is just one of a multitude of Gatsby plugins that can provide data through the GraphQL interface. Any of that data can be used to programmatically create pages.

Other resources

- Example Repository
- Using Gatsby without GraphQL
- CodeSandbox example creating pages from 3rd party data