Using Airtable With Gatsby

Daniel Stout

> I first discovered Airtable in late 2018, and was immediately blown away. For those not already familiar with it yet, Airtable is a cloud-based collaborative database-spreadsheet hybrid application. It offers a flexible and intuitive GUI, reasonable prices (along with a generous free tier), and a full REST API with excellent auto-generated documentation. It also works **great** as a data source for Gatsby.js, and I'm excited to show you how!

stay in the loop with our monthly web dev newsletter:

you@awesome.com

Subscribe!

follow us @alligatorio

site sponsors

SNIPCART

Add custom e-commerce to any site in minutes.

learn more →

about sponsors

In this lesson we are going to create a very basic food truck menu using Airtable, and then display it (almost) effortlessly in a Gatsby website. This article assumes you already have a basic Gatsby.js project set up and you're ready to edit. If you need help getting to that point, try going through the Your First Steps with Gatsby v2 post and then return here afterwards.

Alligator.io recommends ↴

⚛ React for Beginners by Wes Bos

ⓘ About this affiliate link

Airtable Setup

Since this is more of a Gatsby-focused article, we will only cover the general steps involved on the Airtable side of things. You will quickly see that Airtable's UI is **very** intuitive, and the auto-generated documentation is top-notch when needed. I encourage you to take a slow afternoon to explore the features and functionality available with it, though… As you'll discover just how powerful and flexible it can be. (I use it in several personal and client projects, and not just with Gatsby!)

1. Create an Airtable account

The first thing we need to do is create an Airtable account, unless you already have one. Just head over to the Airtable website and click the 'Sign Up' link in the main nav.

Once you're signed up (and/or signed in), you will be at the main dashboard. You're initially provided with a single `workspace`, which holds a collection of demonstration `bases` (a.k.a. databases.) Each base can contain any number of tables, which function like database tables.

Workspaces and bases are free to create, so feel free to create a new empty workspace here if you like. (This is recommended, but not required.)

2. Create a new base

For our food truck menu, we need to create a new base. Let's create one by clicking the gray 'Add a base' button inside the workspace of your choice.

Select the "start from scratch" option when asked, and then in the pop-up we will give our new base a title. (You can also assign a color and glyph for the icon, if you want.) Let's name ours `ChompsFoodTruck`.

> 💡 **Pro Tip:** When working with client projects in Airtable, `always` create a new workspace for each client. Ownership of bases cannot be transferred, but ownership of workspaces can!

3. Rename the default table

With our new base initialized, click on its icon to start configuring it.

You'll notice that the tab at the top is labeled `Table 1`, so we should rename it to something more appropriate. Let's rename it to `Sandwiches` for our purposes.

Since we reference this table name over in Gatsby, I recommend using camelCase or PascalCase naming if your table name needs multiple words, e.g. `SoupsAndSalads`

4. Edit the table

Let's edit the `Sandwiches` table structure to fit our needs:

- `Step 1:` Delete the default `Notes` and `Attachments` columns.
- `Step 2:` Create two new columns, `Description` (type: Single Line Text) and `Price` (type: Currency)
- `Step 3:` Add a few rows of sample data! Use your favorite sandwiches, and make up a price and description for each.

With those steps out of the way, we have a simple `Sandwiches` table set up in our `ChompsFoodTruck` base. We could easily duplicate this to make other menu sections, like sides and/or drinks, but we'll keep it simple for this lesson.

5. Airtable API Credentials

We only have one step left in Airtable: obtaining our API credentials. Making calls to the Airtable API requires both a `base ID` and an `API Key`.

To get the base ID, click the `Help` link next to your profile link (top right), and then click `API documentation` in the dropdown. This will auto-generate fresh documentation for our `ChompsFoodTruck` base in a new browser tab. Then click the `Sandwiches` menu link, and in the example (JavaScript) code on the right you'll see the base ID listed:

```
var base = new Airtable({apiKey: 'YOUR_API_KEY'})
               .base('appABCD12345'); // 👈 nice!
```

To get your API key, just head over to your account overview page. There you'll see a button to (re)generate a new key.

Our Airtable setup is complete! (And that auto-generated documentation was neat, right?!) Let's head back over to our Gatsby project, where we will bring in this data with almost zero effort.

Bringing the Data into Gatsby

We could write some code using the code examples provided by the Airtable documentation, and it's not difficult to work with. However, since we are using Gatsby it's `always` a good idea to look in the plugins section on their official site for something that will help.

Luckily for us, there is an excellent gatsby-source-airtable plugin that we can use! This plugin can fetch rows of data from multiple bases and tables in Airtable, and it automatically converts them all into GraphQL nodes.

1. Installation

The first thing we need to do is install the `gatsby-source-airtable` plugin. Let's stop our development server if it's running, and then at the command prompt:

```
$ yarn add gatsby-source-airtable
```

2. Plugin Configuration

As with all Gatsby plugins, we also need to include it in `gatsby-config.js` and set some configuration options. Let's do that now by inserting it into the `plugins` array, as shown here:

↓ gatsby-config.js

```
// ... siteMetadata above here
plugins: [
  {
    resolve: `gatsby-source-airtable`,
    options: {
      apiKey: `YOUR_AIRTABLE_API_KEY`,
      tables: [
        {
          baseId: `AIRTABLE_BASE_ID`,
          tableName: `Sandwiches`
        },
        // We can add other bases/tables here, too!
        //{
          //baseId: `AIRTABLE_BASE_ID`,
          //tableName: `Sides`
        //}
      ]
    }
  },
  // ... other plugins here
]
```

This is the minimum configuration, and it is all that we need to retrieve our sandwich data. Let's make sure it works by restarting our dev server and then opening GraphiQL in a browser tab. (Typically this URL is: `http://localhost:8000/___graphql`)

You should see two new entries at the top of the `Explorer` menu on the left: `airtable` and `allAirtable`. If you query the data in `allAirtable.nodes`, you should see all of your sandwich data listed. Awesome! 🥪🥪🥪

3. Displaying our data

At this point we now have our Airtable base set up, and we've already got the data in GraphQL nodes available to Gatsby. All that's left is do is query and display it on our site! We could certainly do that by using a page-level query, but for maximum portability let's instead create a new `Menu` component and use a static query within it.

Let's create a new file at `/src/components/Menu.js`, and add the following code:

↓ /src/components/Menu.js

```
import React from 'react';
import { useStaticQuery, graphql } from 'gatsby';

const Menu = () => {
  const data = useStaticQuery(graphql`
    query MenuQuery {
      sandwiches: allAirtable(
        filter: { table: { eq: "Sandwiches" } }
        sort: { fields: data___Price, order: DESC }
      ) {
        nodes {
          data {
            Name
            Price
            Description
          }
          recordId
        }
      }
    }
  `);

  return (
    <div>
      <h3>Sandwiches</h3>

      <ul>
        {data.sandwiches.nodes.map((item, i) => (
          <li key={item.recordId}>
            <p>
              {item.data.Name}, ${item.data.Price}
            </p>
            <p>{item.data.Description}</p>
          </li>
        ))}
      </ul>
    </div>
  );
};

export default Menu;
```

As you can see, we are just mapping over our sandwich data and returning `<li>` elements. Note that we are making use of a newer feature of Gatsby, useStaticQuery, which uses a built-in React Hook to allow GraphQL queries inside any component **at build time.**

> ⚠ Important: This feature requires Gatsby version 2.1.0 or higher, and React 16.8.0 or higher.

Also notice that we're making use of a `filter` option in the query to ensure we are only retrieving data from the `Sandwiches` table. (In this manner, we could create additional queries if we had other menu sections!)

That's it! We can now use our new `Menu` component anywhere in our project, just like any other component. (I would recommend styling it, though!)

Other plugin config options

We've already used the two **required** options within each `tables` object, `baseId` and `tableName`, and those don't really need much explanation. But there are a few other useful options available inside each `tables` object:

- `tableView:` (String) This option allows you to use a custom view that you've set up inside Airtable. (e.g. for highly customized row ordering and/or filtering you've done on the Airtable side.)
- `queryName:` (String) Sometimes you may be using two bases that have tables with the same names. With this option, you can set an alternate name for a table to make GraphQL queries easier to work with.
- `mapping:` (Object) This option lets you map columns to specific data types for Gatsby to transform. It's invaluable for using with markdown data or with image attachments!
- `tableLinks:` (Array of strings) Airtable offers a special column type that links to entries of other tables. With this option, we define these column names to make sure Gatsby retrieves the full data. (Otherwise, it will only fetch each linked item's ID.)

The plugin documentation goes into greater detail about these options, and some fantastic usage examples are available in the plugin's Github repo, including both image processing and markdown processing examples.

Conclusion

This menu example was somewhat basic, but hopefully you've seen that Airtable and Gatsby.js make an incredible duo – especially with help from the gatsby-source-airtable plugin. Don't stop here, though! I definitely encourage you to further explore and tinker around with Airtable's features on your own. Possible ideas:

- Try adding new tables with other menu sections, e.g. `Nachos` and `Drinks`.
- Try creating an `Info` table to store the food truck's email, phone number, logo, etc.
- Try adding images for your menu items, and then use Gatsby's fantastic `Image` component to display them.