Adding an RSS Feed

What is an RSS feed?

An RSS Feed is a standard XML file listing a website's content in a subscribable format, allowing readers to consume your content in news aggregators, also called feed reader apps.

Think of it as a syndicated distribution channel for your site's content.

Install

To generate an RSS feed, you can use the `gatsby-plugin-feed` package. To install this package, run the following command:

```
npm install gatsby-plugin-feed
```

How to use gatsby-plugin-feed

Once installation is complete, you can now add this plugin to your site's config file, like so:

```
module.exports = {
  siteMetadata: {
    siteUrl: `https://www.example.com`,
  },
  plugins: [`gatsby-plugin-feed`],
}
```

Here's an example of how you could implement this plugin with Markdown, but for other sources, you will need a way to uniquely identify content–typically a URL or slug.

```
const { createFilePath } = require(`gatsby-source-filesystem`)

exports.onCreateNode = ({ node, actions, getNode }) => {
  const { createNodeField } = actions
  // highlight-next-line
  if (node.internal.type === `MarkdownRemark`) {
    const value = createFilePath({ node, getNode })
    createNodeField({
```

```
      name: `slug`,
      node,
      value,
    })
  }
}
```

Next run a build (`npm run build`) since the RSS feed generation will only happen for production builds. By default, the generated RSS feed path is `/rss.xml`, but the plugin exposes options to configure this default functionality.

For basic setups with Markdown content like the gatsby-starter-blog, that's all you need! However, you can craft a custom RSS feed schema using custom code in your `gatsby-node.js` and `gatsby-config.js` files.

Customizing the RSS feed plugin

Your content might not fit neatly into the blog-starter scenario, for various reasons like:

- Your content isn't in Markdown so the plugin doesn't know about it
- Your Markdown files have dates in the filenames, for which the slug URLs cause 404s

The good news is you can accommodate these scenarios and more in `gatsby-config.js` and `gatsby-node.js`.

To customize the default feed schema (a.k.a. structure) output by the plugin to work with your website's content, you can start with the following code:

```
module.exports = {
  plugins: [
    {
      resolve: `gatsby-plugin-feed`,
      options: {
        query: `
          {
            site {
              siteMetadata {
                title
                description
                siteUrl
                site_url: siteUrl
              }
            }
          }
        `,
        feeds: [
          {
```

```
            /* highlight-start */
            serialize: ({ query: { site, allMarkdownRemark } }) => {
              return allMarkdownRemark.edges.map(edge => {
                /* highlight-end */
                return Object.assign({}, edge.node.frontmatter, {
                  description: edge.node.excerpt,
                  date: edge.node.frontmatter.date,
                  url: site.siteMetadata.siteUrl + edge.node.fields.slug,
                  guid: site.siteMetadata.siteUrl + edge.node.fields.slug,
                  custom_elements: [{ "content:encoded": edge.node.html }],
                })
              })
            },
            query: `
              {
                // highlight-next-line
                allMarkdownRemark(
                  sort: { order: DESC, fields: [frontmatter___date] },
                ) {
                  edges {
                    node {
                      excerpt
                      html
                      fields { slug }
                      frontmatter {
                        title
                        date
                      }
                    }
                  }
                }
              }
            `,
            output: "/rss.xml",
            title: "Your Site's RSS Feed",
          },
        ],
      },
    },
  ],
}
```

This snippet contains a custom `gatsby-plugin-feed` setup in `gatsby-config.js` to query metadata for your site, like its `title` and `siteUrl`. It also includes a `feeds` array with at least one object containing a GraphQL query and `serialize` method, which allows you to output a custom RSS feed structure.

In this example, the RSS content comes from Markdown files sourced from your site, and queried with the key `allMarkdownRemark` and its associated filters and fields.

The `output` field in your feed object allows you to customize the filename for your RSS feed, and `title` for the name of your site's RSS feed.

By default, feed is referenced in every page. You can customize this behavior by providing an extra field `match` of type `string`. This string will be used to build a `RegExp`, and this regular expression will be used to test the `pathname` of current page. Only pages that satisfied the regular expression will have feed reference included.

If your site has none-English link (or none-ASCII link), you may need to encode URI in advance. You can use the build-in function `encodeURI(string)` for your link:

```
serialize: ({ query: { site, allMarkdownRemark } }) => {
  return allMarkdownRemark.edges.map(edge => {
    return Object.assign({}, edge.node.frontmatter, {
      description: edge.node.excerpt,
      date: edge.node.frontmatter.date,
      // highlight-next-line
      url: encodeURI(site.siteMetadata.siteUrl + edge.node.fields.slug),
      guid: site.siteMetadata.siteUrl + edge.node.fields.slug,
      custom_elements: [{ "content:encoded": edge.node.html }],
    })
  })
},
```

To see your feed in action, run `gatsby build && gatsby serve` and you can then inspect the content and URLs in your RSS file at `http://localhost:9000/rss.xml`. You can check out the validation of your RSS feed on W3C Feed Validation Service.

> NOTE: if your blog has custom permalinks, such as links with or without dates in them, you may need to customize `gatsby-node.js` to output the correct URLs in your RSS feed. Get in touch with us if you need any help!

Syntax for iTunes RSS blocks

If creating a RSS feed for a podcast you probably will want to include iTunes RSS blocks. They take the format of `itunes:author` which GraphQL does not read. Here's an example of how to implement iTunes RSS blocks using this plugin:

```
module.exports = {
  plugins: [
```

```
    {
      resolve: `gatsby-plugin-feed`,
      options: {
        query: `
          {
            site {
              siteMetadata {
                title
                description
                siteUrl
                site_url: siteUrl
              }
            }
          }
        `,
        /* highlight-start */
        setup: options => ({
          ...options,
          custom_namespaces: {
            itunes: 'http://www.itunes.com/dtds/podcast-1.0.dtd',
          },
          custom_elements: [
            { 'itunes:author': 'Michael Scott' },
            { 'itunes:explicit': 'clean' },
          ],
        }),
        /* highlight-end */
        feeds: [
          {
            ...
          },
        ],
      },
    },
  ],
}
```

Happy blogging!

With the Gatsby feed plugin, you can share your writing easily with people subscribed through RSS readers like Feedly or RSS Feed Reader. Now that your feed is set up, you won't really have to think about it; publish a new post, and your RSS feed will automatically update with your Gatsby build. Voilà!

More resources

Jason Lengstorf and Amberley Romo livestream building an RSS feed powered podcast site.