Create REST API documentation

Last Modified on 03/14/2024 11:49 am EDT

Learn how to create REST API documentation using KnowledgeOwl and Redoc.

REST API documentation in KnowledgeOwl

When creating API documentation, very often you'll want to be able to automatically generate docs from a specification file or from the code. KnowledgeOwl does not have built-in support for this.

However, it is possible to integrate with other tools that do. This guide covers how to set up REST API documentation in KnowledgeOwl using an OpenAPI specification file (also known as Swagger spec) and Redoc (using its HTML element method). It is the same technique we use for our own endpoint reference docs.

Prerequisites

- You need to have an OpenAPI spec file that is compatible with Redoc. If your spec file is version 3.0, make sure to use Redoc 2.0. Refer to Redoc's version guidance for more information.
- If you want to customize the look and feel of your API docs, you will need some knowledge of HTML and CSS and how to use your browser's developer tools.
- You should familiarize yourself with ReDoc's configuration options.

Create your API documentation

1. Create a new article, or a custom content category.
2. In **Display Settings**, select **Remove feedback ability**, **Remove comment ability**, and **Remove "Download to PDF" icon**.
3. Host your spec file, choose your Redoc source, and make a note of the URLs: your specification file, and the Redoc JavaScript file, must be hosted somewhere so that your page can load them. You can choose to host the spec file anywhere you want, and you can use Redoc's own recommended CDN to load the JavaScript: `https://cdn.jsdelivr.net/npm/redoc@latest/bundles/redoc.standalone.js`
   However, you can also use the KnowledgeOwl **File Library** to host both files. Refer to Files and images for information on uploading files and getting the URLs.

   When using the KnowledgeOwl file library, you need to edit the URLs. The full URL of a file has this format: https://<random-id>.cloudfront.net/app/image/id/<id>/n/<filename> Remove everything before 'app', giving you a URL with this format:

/app/image/id/<id>/n/<filename>.

4. Create a snippet and add the following in the **Code Editor**view. Replace 'url-to-your-spec-file' with the URL of your OpenAPI specification file, and '' with the URL to redoc.standalone.js (this may be your own copy, or a call to a CDN).

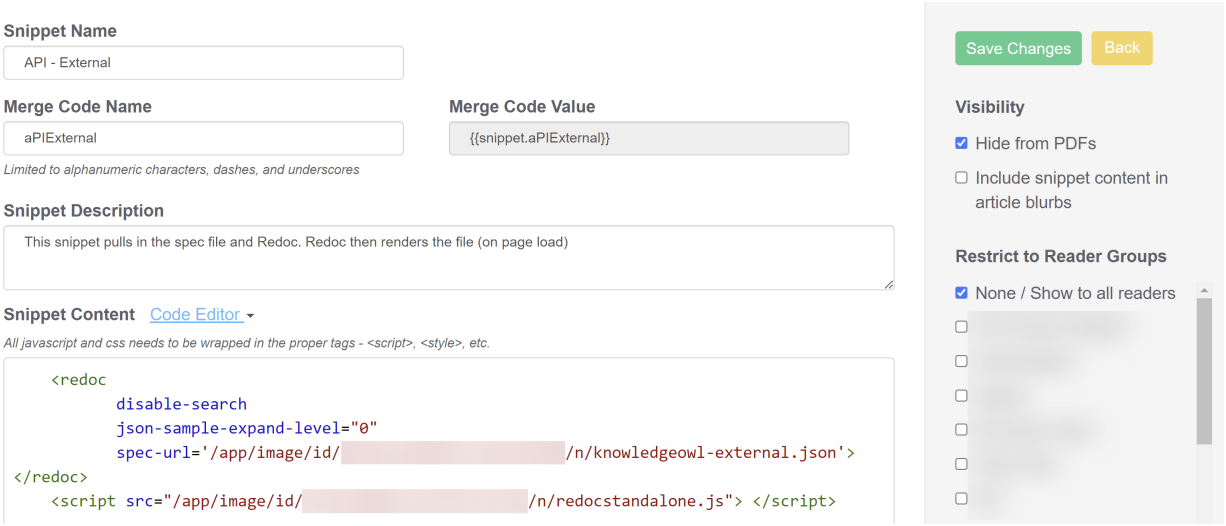
```
<redoc disable-search spec-url='url-to-your-spec-file'></redoc>
<script src=""></script>
```

This adds the `<redoc>` custom element, which contains the Redoc configuration options and spec file URL. It then loads the Redoc JavaScript. When a reader visits the article, the script runs and renders the spec file.

We recommend including the `disable-search` option. You can choose to add other configuration options to suit your needs.

5. Under **Visibility**, select **Hide from PDFs**.

6. **Save** the snippet. It should now look like this:



7. Insert the snippet into the article or category you created in step 1.

8. Set the **Publishing Status** to **Published**.

9. Select **Save**. You can now view the documentation.

10. *Optional:* you can add custom styling. Some changes, such as expand/collapse behaviors, hiding elements of the Redoc theme, and so on, should be controlled using the Redoc configuration options. You can also add custom CSS to change things like colors. You could add your custom CSS to the snippet you created in step 4, or in its own snippet.

Custom styles

There are two ways to customize the look and feel of Redoc's output:

- Redoc's theming options. You can set these on the `<redoc>` element. Note that the available options for the open source community edition are limited. As of 24 September 2021, there seems to be a bug affecting this functionality. You can follow the issue on GitHub. If you use this option, you may still need some custom CSS to override your knowledge base's default styles.
- Custom CSS, added to the article or category. This is the option we chose for our own docs. It gave us more fine-grained control of the look and feel, and avoided issues caused by the possible bug linked above.

Redoc's class names change with every new release. If you go with the custom CSS option, make sure you are hosting your own copy of redoc.standalone.js, and be aware that if you upgrade the Redoc version, you will need to fix your styles.

To customize the look and feel of our own docs, we created a snippet to hold the CSS. This allows us to reuse it if we wish. We then inserted the CSS snippet into our API endpoint article.

You can find out what CSS classes to target by using your browser's development tools to inspect the page source (you will need to create the documentation first, so that you can inspect it).

The following sections describe some of our custom CSS, as an example to get you started on your own custom styling.

Overriding KnowledgeOwl styles

The first thing we needed to do was override some of the styling inherited from the knowledge base theme. These changes ensure the API docs can use the full width of the screen, remove unwanted whitespace (padding and margin) around the edges, and hide some knowledge base elements that we didn't want on the page, such as the header and breadcrumbs. Your knowledge base theme may use different class names.

```
/* Allow the API docs to use the full width of the browser */
.ko-content-cntr, .hg-article {
 max-width: 100% !important;
}

/* Remove padding and margins */
.hg-minimalist-theme .hg-article-body {
  padding-bottom: 0;
}

.hg-article-body p {
 margin: 0;
}

.ko-site-footer {
 padding: 0;
}

#ko-article-cntr {
 padding-top: 0;
}

/* Hide the back to top button */
.back-to-top {
 display: none !important;
}

/* Hide the header, breadcrumbls, and article footer */
.hg-article-header, .hg-breadcrumbs, .hg-article-footer {
 display: none;
}
```

Overriding Redoc styles

Next, we targeted various Redoc classes to add our brand colors.

```
/* Right hand column background color */
.iQAzHu, .dAURiS {
 background-color: #1D284F;
}

/* Expanding text buttons under 'Responses' - brand colors and improved alignment */
/* 200 */
.jekxwK {
 background-color: #F8FBF9;
 display: flex;
 flex-direction: row;
 align-items: flex-end;
 justify-content: flex-start;
}
.jekxwK > * {
 margin: 2px;
}

/* Error */
.epQWfk {
 background-color: #F4E2E2;
 border: 1px solid #E6ADA9;
 color: #FD4339;
 display: flex;
 flex-direction: row;
 align-items: flex-end;
 justify-content: flex-start;
}
.epQWfk > * {
 margin: 2px;
}

/* Verb labels in left and right hand column - colors */
.bGXNhC, .ifwede.post {
 background-color: #017AFF;
}
.gPqqEc, .ifwede.get {
 background-color: #5c995b;
}
.hAtMod, .ifwede.put {
 background-color: #9E57B4;
}
.eZVHeg, .ifwede.delete {
 background-color: #FD4339;
}
```

Limitation: content security policy header

If you have content security policy headers enabled in your knowledge base security settings, you cannot host the files in the KnowledgeOwl file library. The security settings prevent loading the JSON or YAML spec file.

You can still load the files from an external server, but must make sure they load over HTTPS. For example, Redoc's own example will work, so long as you update the URL to the example spec to use HTTPS:

```
<redoc spec-url='https://petstore.swagger.io/v2/swagger.json'></redoc>
<script src="https://cdn.jsdelivr.net/npm/redoc@latest/bundles/redoc.standalone.js"> </script>
```

Understand Redoc's standalone API docs

Redoc offers several ways to create API docs. For our own documentation, we use a custom HTML element and Redoc's 'redoc.standalone.js' script. This works by loading the spec file and any configuration options set in the custom element, then assembling the documentation and applying the Redoc theme. It does this every time the page loads.

There are pros and cons to this approach. It is relatively simple to implement, requires no additional software or build processes, and the look and feel are fairly easy to customize by adding additional CSS. However, be aware that large spec files will take some time to load and build, resulting in a loading screen.

An alternative method is to generate the documentation using the Redoc CLI. You can then upload the generated HTML into the file library (or host it yourself), and include it in your knowledge base using a URL redirect category or article. This removes the build step on page load, meaning faster load times and no loading screen. However, it makes it harder to apply custom styling (you are limited to the theming options supported by Redoc, which are minimal for the open source community edition). It also adds a build process that you must manage, for example installing Node.js and the Redoc CLI on your computer.

Next steps

This article has described one method of creating REST API documentation from an OpenAPI spec in KnowledgeOwl. If you want to use it, it's worth spending time reading Redoc's README and documentation. You can also view our own endpoint docs as an example.

If you don't use OpenAPI as your API specification, consider looking for other tooling similar to Redoc: if it can either pre-generate HTML, or provides a way to generate in the browser, then it may be possible to use it alongside KnowledgeOwl. We'd love to hear from you if you discover any interesting tools!