@push.rocks/smartdocumentation

A tool for converting git directory structures into navigable documentation sites.

readme.md for @push.rocks/smartdocumentation

a tool for mapping git directories to documentation sites

Install

To install `@push.rocks/smartdocumentation`, you need to have Node.js and npm installed on your machine. Once you have those, run the following command in your terminal:

```
npm install @push.rocks/smartdocumentation --save
```

This will add `@push.rocks/smartdocumentation` to your project's dependencies.

Usage

The `@push.rocks/smartdocumentation` package allows you to map your git directory structures into documentation sites efficiently. It leverages TypeScript and a set of sophisticated tools under the hood such as `@push.rocks/smartfile`, `@push.rocks/smartmarkdown`, and `@tsclass/tsclass` to manage and generate documentation content dynamically.

Setting Up

First, ensure you are using TypeScript and have it configured in your project. You can include `@push.rocks/smartdocumentation` in your TypeScript file with the following import statement:

```
import { DocumentationDirectory } from '@push.rocks/smartdocumentation';
```

Creating a Documentation Directory

To use the functionality provided by this package, you need to create an instance of the `DocumentationDirectory`. This instance will represent a specific directory containing Markdown files that you intend to use for documentation.

Here's how you can create a `DocumentationDirectory` instance:

```
import { DocumentationDirectory } from '@push.rocks/smartdocumentation';

const myDocDir = new DocumentationDirectory({
  pathArg: './path/to/your/documentation',
});
```

Make sure to replace `'./path/to/your/documentation'` with the actual path to your documentation directory.

Reading the Directory

After creating an instance of `DocumentationDirectory`, you can read the directory to process the Markdown files within:

```
await myDocDir.readDirectory();
```

This method asynchronously processes each Markdown file, extracting information and preparing them for further actions like sending them as a documentation set or rendering.

Processing and Utilizing Documentation

Once the directory is read, you have several options on how to use the processed documentation. For illustration, let's assume you want to print the titles of all articles in your console:

```
for (const article of myDocDir.articles) {
  console.log(article.title);
}
```

Sending Documentation to a Destination

While the base package provides you with the tools to read and process documentation, sending this documentation to a specific target or rendering it into a website would require you to implement or use further tools or methods, tailored to your specific needs.

Example Use Case

Imagine you are managing a project documentation stored in Markdown files within a git repository. You want to create a documentation site that reflects the structure and content of these files. With `@push.rocks/smartdocumentation`, you can automate the collection and processing of these files, preparing them for a static site generator or a custom rendering engine to display them online.

You would start by organizing your Markdown files in a clear directory structure within your project. Then, use `@push.rocks/smartdocumentation` to create instances of `DocumentationDirectory` for each directory you intend to document. After processing these directories, you would extract the necessary metadata, content, and structure to feed into your site generator or rendering engine, automating the documentation site's update process as your project evolves.

Conclusion

The `@push.rocks/smartdocumentation` package provides a powerful base for handling the conversion of structured, Markdown-based documentation into a format ready for online presentation. With a little setup and some custom tooling around your specific output needs, it can significantly streamline the documentation process for projects of any size.

License and Legal Information

Trademarks

Company Information

Task Venture Capital GmbH
Registered at District court Bremen HRB 35230 HB, Germany

For any legal inquiries or if you require further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.