@push.rocks/smartswagger

A Swagger toolkit for working with Swagger files, including merging documents and serving APIs with UI.

readme.md for @push.rocks/smartswagger

a swagger tookit for working with swagger files

Install

To install `@push.rocks/smartswagger`, use npm:

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

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

Usage

`@push.rocks/smartswagger` provides a comprehensive toolkit for working with Swagger (now known as OpenAPI) files in a TypeScript and ESM syntax environment. Below, we'll dive into several scenarios covering how you can utilize `@push.rocks/smartswagger` effectively in your projects.

Creating a New `Smartswagger` Instance

Start by creating a new instance of `Smartswagger`. You can do this from a URL pointing to a Swagger file or by manually specifying Swagger document properties.

From an External Swagger File URL

```
import { Smartswagger } from '@push.rocks/smartswagger';

async function loadSwaggerFromUrl() {
  const swaggerInstance = await
Smartswagger.createFromUrl('https://example.com/path/to/swagger.json');
  // Now you have a Smartswagger instance created from an external Swagger file
```

```
}
```

Creating a New Swagger Document

```
import { Smartswagger } from '@push.rocks/smartswagger';

async function createNewSwaggerDoc() {
  const swaggerInstance = await Smartswagger.createNew('My API Documentation');
  // Now you have a new Smartswagger instance with basic setup
}
```

Merging Swagger Documents

One of the powerful features of `@push.rocks/smartswagger` is the ability to merge multiple Swagger documents into a single one. This is particularly useful when working with microservices or splitting your API documentation across several files for better organization.

Merging Documents from URLs

Suppose you have several Swagger documents for different parts of your API hosted at various URLs. You can merge them into a single document like so:

```
import { Smartswagger } from '@push.rocks/smartswagger';

async function mergeDocuments() {
  const mainApiDoc = await Smartswagger.createNew('Combined API Documentation');

  await mainApiDoc.mergeManyDocumentsFromUrl([
    { url: 'https://example.com/path/to/users/api/swagger.json' },
    { url: 'https://example.com/path/to/products/api/swagger.json', basePath: '/products' },
    // Include as many as needed
  ]);

  // Now mainApiDoc contains a merged version of all specified Swagger documents
}
```

Adding Servers and Security Schemes

You can add servers and security schemes to your Swagger document as follows:

Adding a Server

```
import { Smartswagger } from '@push.rocks/smartswagger';

async function addServer() {
  const swaggerInstance = await Smartswagger.createNew();

  await swaggerInstance.addServer({
    url: 'https://api.example.com/',
    description: 'Main API Server'
  });

  // Your Swagger instance now includes the server information
}
```

Merging Security Schemes to Routes

```
import { Smartswagger, openapiTypes } from '@push.rocks/smartswagger';

async function addSecuritySchemes() {
  const swaggerInstance = await Smartswagger.createNew();

  const securityScheme: openapiTypes.OpenAPIV3.SecuritySchemeObject = {
    type: 'http',
    scheme: 'bearer',
    bearerFormat: 'JWT'
  };

  swaggerInstance.mergeComponentToRoutes({ includeGlobArray: ['**'], excludeGlobArray: [] }, {
securitySchemes: { bearerAuth: securityScheme } });

  // Now, all routes in your Swagger document include the specified security scheme
}
```

Integrating with Express and Serving Your API Documentation

`@push.rocks/smartswagger` integrates seamlessly with Express, allowing you to serve your API documentation using Swagger UI or ReDoc.

```
import { Smartswagger } from '@push.rocks/smartswagger';
import * as smartexpress from '@pushrocks/smartexpress';

async function serveSwaggerUI() {
  const swaggerInstance = await Smartswagger.createNew('My API Documentation');
  // proceed with merging documents, adding servers, etc., as shown in previous examples

  const expressServer = new smartexpress.Server({ cors: true });

  // Serve Swagger UI
  expressServer.addRoute('/api/documentation', new smartexpress.Handler('ALL',
swaggerInstance.getSlashApiUiMiddleware()));

  // Optionally, serve documentation using ReDoc
  expressServer.addRoute('/api/redoc', new smartexpress.Handler('ALL',
swaggerInstance.getSlashRedocMiddleware()));

  // Serve the raw Swagger JSON
  expressServer.addRoute('/api/swagger.json', new smartexpress.Handler('ALL',
swaggerInstance.getSlashApiSchemaMiddleware()));

  // Start the server
  await expressServer.start(3000);
}
```

Conclusion

`@push.rocks/smartswagger` offers a robust toolkit for working with Swagger documents in TypeScript, making it easier to build, merge, and serve comprehensive API documentation. Whether you're dealing with a single document or coordinating documentation across multiple services, `@push.rocks/smartswagger` provides the features necessary to streamline these processes.

License and Legal Information

This repository contains open-source code that is licensed under the MIT License. A copy of the MIT License can be found in the license file within this repository.

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH and are not included within the scope of the MIT license granted herein. Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines, and any usage must be approved in writing by Task Venture Capital GmbH.

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.