@push.rocks/smartvhost

A module for creating a virtual host (vhost) layer to manage multiple SmartExpress instances.

- readme.md for @push.rocks/smartvhost

readme.md for @push.rocks/smartvhost

a module creating a vhost layer in order to maintain several smartexpress instances

Install

To install `@push.rocks/smartvhost`, use the following npm command:

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

This will add smartvhost as a dependency to your project and you are ready to use it to manage multiple virtual hosts within your applications.

Usage

`@push.rocks/smartvhost` simplifies the handling of multiple `smartexpress` instances or other target types by providing a virtual host (vhost) layer. This allows for differentiated handling of requests based on the hostname, among other criteria. Utilizing this module enables efficient scalability and organization of your web server components in a Node.js environment.

Below, you'll find comprehensive examples and explanations to help you get started and leverage the full spectrum of functionalities offered by smartvhost.

Importing and Initial Setup

Firstly, let's look into importing and setting up smartvhost in your project using modern ESM syntax and TypeScript.

```
import { SmartVHost } from '@push.rocks/smartvhost';


const mySmartVHost = new SmartVHost();
```

Upon instantiation, `SmartVHost` begins without any configured virtual hosts. To make it functional, you'll need to define and set configurations according to your needs.

Configuring Virtual Hosts

Virtual host configurations are crucial for directing traffic appropriately. smartvhost supports several types of configurations, which include directing traffic to local folders, specific IP and port combinations, local ports, or even other domains. Below is how to define and apply these configurations:

```
import { IVHostConfig } from '@push.rocks/smartvhost';

// Define an array of virtual host configurations
const vHostConfigs: IVHostConfig[] = [
  {
    hostName: 'example.com',
    type: 'folder',
    target: '/path/to/your/site',
    privateKey: '<SSL_PRIVATE_KEY>', // needed for HTTPS
    publicKey: '<SSL_PUBLIC_KEY>', // needed for HTTPS
  },
  {
    hostName: 'api.example.com',
    type: 'localPort',
    target: '3002',
    privateKey: '<SSL_PRIVATE_KEY>',
    publicKey: '<SSL_PUBLIC_KEY>',
  },
  {
    hostName: 'another-domain.com',
    type: 'ipAndPort',
    target: '192.168.1.100:80',
    privateKey: '<SSL_PRIVATE_KEY>',
    publicKey: '<SSL_PUBLIC_KEY>',
  }
];

// Apply the configurations to the smartvhost instance
mySmartVHost.setVHostConfigs(vHostConfigs);
```

Starting and Stopping smartvhost

To activate the virtual host layer, simply start your smartvhost instance. Similarly, you can stop it when necessary:

```
// To start
await mySmartVHost.start();

// When needed, stop
await mySmartVHost.stop();
```

Complete Example

Combining all the steps above, here's a complete example showcasing how to utilize `@push.rocks/smartvhost` in managing different vhosts for your application:

```
import { SmartVHost, IVHostConfig } from '@push.rocks/smartvhost';

async function setupSmartVHost() {
  const mySmartVHost = new SmartVHost();
  const vHostConfigs: IVHostConfig[] = [
    {
      hostName: 'example.com',
      type: 'folder',
      target: '/path/to/site',
      privateKey: '<SSL_PRIVATE_KEY>',
      publicKey: '<SSL_PUBLIC_KEY>',
    },
    // Include other configurations as needed
  ];

  mySmartVHost.setVHostConfigs(vHostConfigs);

  await mySmartVHost.start();
  console.log('SmartVHost started and serving configurations.');

  // When you're done, don't forget to stop it
  // await mySmartVHost.stop();
}
```

```
setupSmartVHost().catch(console.error);
```

This guide and examples should serve as a foundation to integrate and fully utilize smartvhost in your Node.js projects, thereby enhancing your web server's flexibility and efficiency in handling multiple domains or applications.

For advanced use cases, refer to the module's typedoc documentation and explore the extensive API options provided for fine-tuned control over your virtual host configurations and behaviors.

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.