@design.estate/navigation

navigation for complex webapps

readme.md for @design.estate/navigation

@designestate/navigation

A module for opinionated navigation abstraction.

Install

To use `@designestate/navigation` in your project, run:

```
npm install @designestate/navigation --save
```

This will install the package and add it to your project's dependencies.

Usage

The `@designestate/navigation` module is designed to simplify the creation and management of navigational structures in your application. It provides a way to abstract navigation into manageable components, allowing for a more modular and maintainable codebase. Below are examples and explanations on how to utilize this module effectively in your projects.

Getting Started

First, ensure that you import the necessary classes from the module:

```
import { NavigationManager, NavigationEntry } from '@designestate/navigation';
```

Creating a Navigation Manager

The `NavigationManager` acts as the central point for managing navigation entries and can be linked with other navigation managers to create a more complex navigation structure. Here's how you can initialize a new `NavigationManager`:

```
const myNavigationManager = new NavigationManager();
```

Adding Navigation Entries

`NavigationEntry` instances represent individual navigational elements. These can be pages, links, or any other components that you navigate to in your application. You can add navigation entries to the navigation manager like this:

```
import { NavigationEntry } from '@designestate/navigation';

const dashboardNavEntry = new NavigationEntry();
dashboardNavEntry.level = 0;
dashboardNavEntry.group = 'main';
dashboardNavEntry.groupRanking = 1;
dashboardNavEntry.iconUrl = 'path/to/icon/url';
dashboardNavEntry.id = 'dashboard';
dashboardNavEntry.name = 'Dashboard';
dashboardNavEntry.callBack = (idArg?: string) => {
  console.log(`Navigating to ${idArg}`);
};

myNavigationManager.addNavigationEntry(dashboardNavEntry);
```

Compiling Navigation

Once you have added all necessary navigation entries, you can compile the navigation structure. This process transforms the added entries into a `CompiledNavigation` instance, which can be used to render navigational components or manage routing:

```
myNavigationManager.compile().then(compiledNavigation => {
  // Use the compiledNavigation instance as needed
});
```

Observing Navigation Changes

`NavigationManager` provides an observable you can subscribe to for reacting to navigation changes. Whenever the navigation structure is compiled or updated, the observable emits a `CompiledNavigation` instance:

```
myNavigationManager.compiledNavigationObservable.subscribe(compiledNavigation => {
  // Handle navigation change
});
```

Advanced Usage

Hierarchical Navigation

For complex applications, you might need hierarchical navigation. You can add a `NavigationManager` instance to another `NavigationManager`, enabling hierarchical navigation management:

```
const subNavigationManager = new NavigationManager();
// Configure subNavigationManager

myNavigationManager.addNavigationManager(subNavigationManager);

// When you compile myNavigationManager, it will also include navigation entries from
subNavigationManager
```

Dynamic Navigation Entries

Navigation entries can be dynamically added or removed based on application state or user permissions. This allows for a flexible and adaptive navigation structure:

```
const adminNavEntry = new NavigationEntry();
// Configure adminNavEntry for admin level access

if (user.isAdmin) {
  myNavigationManager.addNavigationEntry(adminNavEntry);
}

// You can also remove entries dynamically if needed
```

Conclusion

The `@designestate/navigation` module offers a powerful and flexible way to manage navigation in your applications. By abstracting navigation into manageable components, it facilitates better organization and maintenance of your codebase. With the ability to create hierarchical and dynamic navigations, this module can cater to a wide range of application structures and scenarios.

Remember, the above examples are just the starting point. Explore the module further to harness its full capabilities in building intuitive and maintainable navigation structures for your applications.

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.