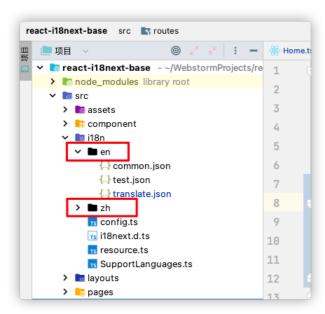


# 1. 语言包

#### 1-1. 语言



这些在后面切换语言是能够用着

#### 1-2. 命名空间

指的是文件名(不包含 .json ),后面在组件中使用,会用到这个概念

文件名	命名空间
translate.json	translate
common.json	common

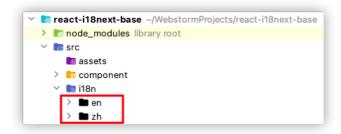
```
react-i18next-base src i18n en {..} common.json
1
   > node_modules library root
                                                          "Save": "Save",
り 拉取端状
   ✓ 🔯 src 🚺
                                                          "Discard": "Discard",
     > En component
                                                          "Confirm": "Confirm",
      i18n2
                                                          "Cancel": "Cancel",
                                                   5
     3 ■ en 

{-} common.json 

{-} translate.json
                                                          "Delete": "Delete",
                                                   7
                                                          "Remove": "Remove",
                                                         "Close": "Close",
                                                   8
           ← common.json
                                                          "Case_N": "Case {{num}}}"
                                                  9
           {→} translate.json
         s config.ts
                                                  10
                                                        ₽}
         i18next.d.ts
         resource.ts
         supportLanguages.ts
      > 🛅 layouts
      > pages
      > Is routes
      > 📘 styles
      > 🔝 types
      > ា utils
       🔐 favicon.svg
        🥮 main.tsx
       vite-env.d.ts
      .editorconfig
```

#### 1-3. 新增命名空间

注意: 下述 {local\_code} 指的是不同语言的文件夹,通常我们把 en 作为**主语言**,所以添加命名空间在 en 文件夹 下面添加即可



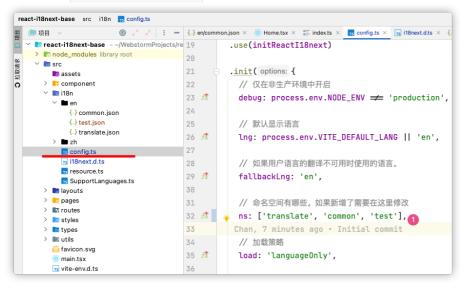
#### 具体流程

- 1. 在 /src/i18/{local\_code}/ 文件夹下新增 .json 文件, 比如 test.json
- 2. 修改文件 /src/i18n/i18next.d.ts 中的接口 I18NResourcesType 添加新的属性定义

```
react-i18next-base src i18n 📆 i18next.d.ts
                     ⊚ 🦿 🎤 🕴 = {{} en/common.json × ∰ Home.tsx × 🏗 index.ts × 📆 config.ts × 📆 i18next.d.ts × {{}} zh/common.json ×
                                   rojects/re 1 jimport 'react-i18next';
2 import translate from './en/translate.json';
      node_modules library root
    ∨ 🛅 src
                                                  import common from './en/common.json';
        assets
                                         4 pimport test from './en/test.json'; 1
                          type Translate = typeof translate;

type Common = typeof common:
             sype Common = typeof common type Test = typeof test; 2
        > = zh
        is i18next.d.ts
resource.ts
SupportLanguages.ts
                                         10 export type {Translate}
                                                 Chan, Moments ago · Initial commit
                                         12 export interface I18NResourcesType {
        layouts
                                                translate: Translate;
        pages
                                         13
                                                    common: Common
                                         15
                                                   test: Test 🔞
      > 📗 utils
                                         16
        a favicon.svg
                                         17
         main.tsx
      vite-env.d.ts
seditorconfig
                                         18
                                                 // TS 定义命名空间, 如果是 JS 这个不需要去定义
                                               declare module 'react-i18next' {
interface CustomTypeOptions {
                                         19
      +†∔ .env
                                         20
      env.development.local
                                                    defaultNS: 'translate';// 默认使用的命名空间
nsSeparator: ':';// 命名空间分隔符
keySeparator: '.';//
        .env.example
                                         21
                                         22
                                         23
      index.html
      package.json
                                                      resources: I18NResourcesType
                                         25
      arn.lock
```

3. 修改文件 /src/il8n/config.ts



### 2. 函数组件中使用 react-i18next

```
1 function Case2() {
   // 这种方式可以传入多个
    // const {t} = useTranslation(['translate'])
4
5
     const {t} = useTranslation('translate')
6
    return (
     <div>
8
      <DisplayText size="large">
9
       <CodeShow>t('Case2')</CodeShow>: 
{t('Case2')}
      </DisplayText>
     </div>
14 )
15 }
```

#### 3. 类组件中使用 react-i18next

```
1 import React from "react";
2 import {WithTranslation} from "react-i18next";
4 // 注意这里,IProps 接口继承 WithTranslation 并指定命名空间
5 // 下述是表示指定了使用 translate 命名空间,以获得 IDEA 参数补全
6 interface IProps extends WithTranslation<'translate'>{
7 }
8
9 export class Home extends React.Component<IProps, any> {
10 render() {
       // 看这里
         const {t} = this.props
       // 使用
14
         return (<h1>{t('Welcome')}</h1>)
    }
16
17 }
18
```

## 4. Trans 组件的使用

## 5. 设置语言

这是我自己封装的一个工具类

```
import i18next from "@/i18n/config";

const local = 'zh'

i18next.changeLanguage(local)
```

### 6. 列表渲染

语言包:

```
1 {
2  "AnimalList": [
3   "I like Panda",
4   "I like Monkey",
5   "I like Pig"
6  ]
7 }
```

代码:

```
t('Case6'): Case 6: List render

I like Panda
I like Monkey
I like Pig
```

# 7. 单复数

语言包:

```
1 {
2    "Case7": {
3         "apple": "I have {{count}}} apple",
4         "apple_other": "I have <b>{{count}}</b> apples"
5    }
6 }
```

代码:

```
1 function Case7() {
  return (
2
     4
      <
5
        <Trans ns="translate" i18nKey="Case7.apple" count={-2}>
6
         I have {{count: -2}} apple
        </Trans>
8
     9
      <
       <Trans ns="translate" i18nKey="Case7.apple" count={-1}>
         I have {{count: -1}} apple
        </Trans>
     14
     <
16
       <Trans ns="translate" i18nKey="Case7.apple" count={0}>
18
         I have {{count: 0}} apple
19
        </Trans>
     <
       <Trans ns="translate" i18nKey="Case7.apple" count={1}>
24
         I have {{count: 1}} apple
       </Trans>
26
      28
       <
```

#### 运行结果:

```
CASE 7
I have -2 apples
I have -1 apple
I have 0 apples
I have 1 apple
I have 2 apples
```

### 8. 非组件中使用翻译

通常我们会在如 API 请求完成需要多语言显示如操作成功之类的

```
1 import i18next from "@/i18n/config";
2
3 /**
4 * 基础使用
5 * @constructor
6 */
7 export function TestUtil_1() {
8 return i18next.t('common:Save')
9 }
11 /**
12 * 返回 object 内容
13 * @constructor
14 */
15 export function TestUtil_2() {
16 return i18next.t('translate:AnimalList', {returnObjects: true})
17 }
18
19 /**
20 * 使用变量
21 *
22 * @param count
23 * @constructor
24 */
25 export function TestUtil_3(count: number = 1) {
26 return i18next.t('common:Case_N', {replace: {num: count}})
27 }
28
```

JSX 代码:

## 9. 🔔 注意事项

1. 对于一些组件并在**非切换路由**下不会刷新组件,导致切换语言时页面语言并不会改变,所以有 2 种方案

a. 切换语言后,主动**刷新页面** 

b. 监听语言切换事件,主动**刷新组件** (如改变 state 等)



Rn-刷新组件Render(重新渲染)... 开发过游戏的都应该很清楚, "···· https://www.jianshu.com/p/63!

#### 大概介绍下第 b. 种解决方案:

这只是**提供一个解决思路**,实际上我这个例子你**不添加监听器**也可以正常切换语言显示

注意看: componentDidMount 和

componentWillUnmount

```
1 import React from "react";
2 import {LanguageUtils} from "@/utils/languages";
3 import {IListenerNameEnum} from "@/utils/languages.d";
5 export default class extends React.Component<any, any> {
6 state = {
    load_time: 0
8
9
10 // 组件挂载,添加监听器
11 componentDidMount() {
    // 注意第 1 个参数,需要在 @/utils/languages.d 定义的枚举
    LanguageUtils.addListener(IListenerNameEnum.FirstListener, () => {
     // 更新 state
14
      this.setState({
         load_time: new Date().getTime
16
     })
     })
18
19
    // 组件销毁,一定要移除监听器!!!!!
    // 组件销毁,一定要移除监听器!!!!!
    // 组件销毁,一定要移除监听器!!!!!
24 componentWillUnmount() {
    LanguageUtils.removeListener(IListenerNameEnum.FirstListener)
26 }
28
```