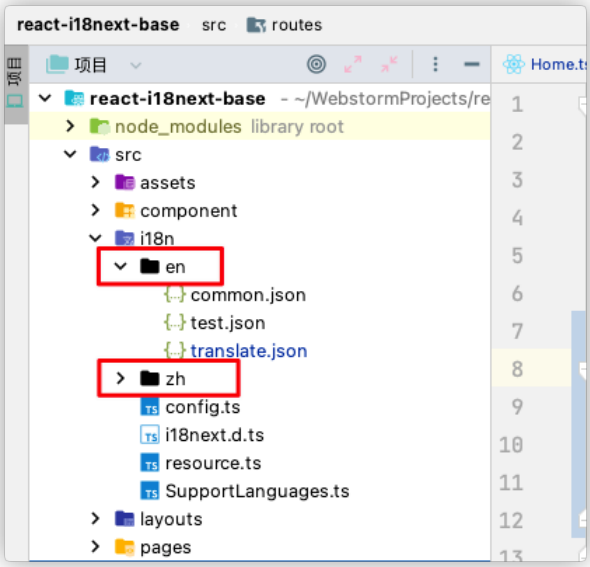
链接
https://react.i18next.com/

1. 语言包

1-1. 语言

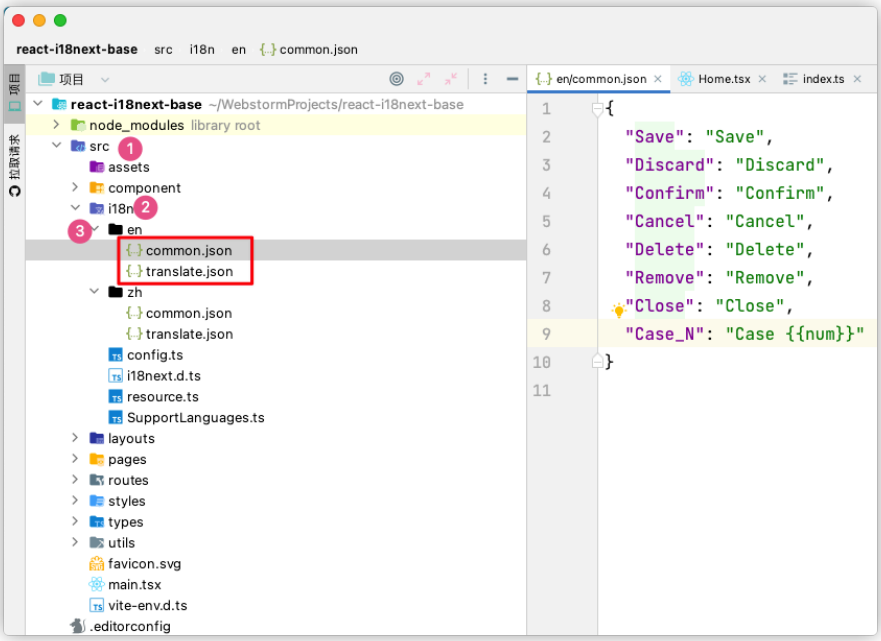


这些在后面切换语言是能够用着

1-2. 命名空间

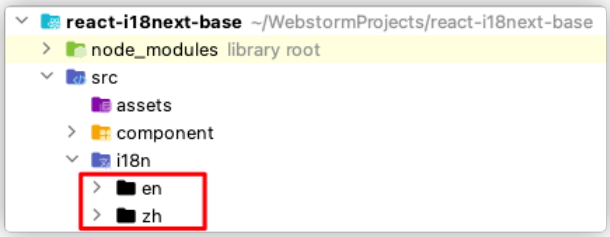
指的是文件名（不包含 `.json` ），后面在组件中使用，会用到这个概念

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



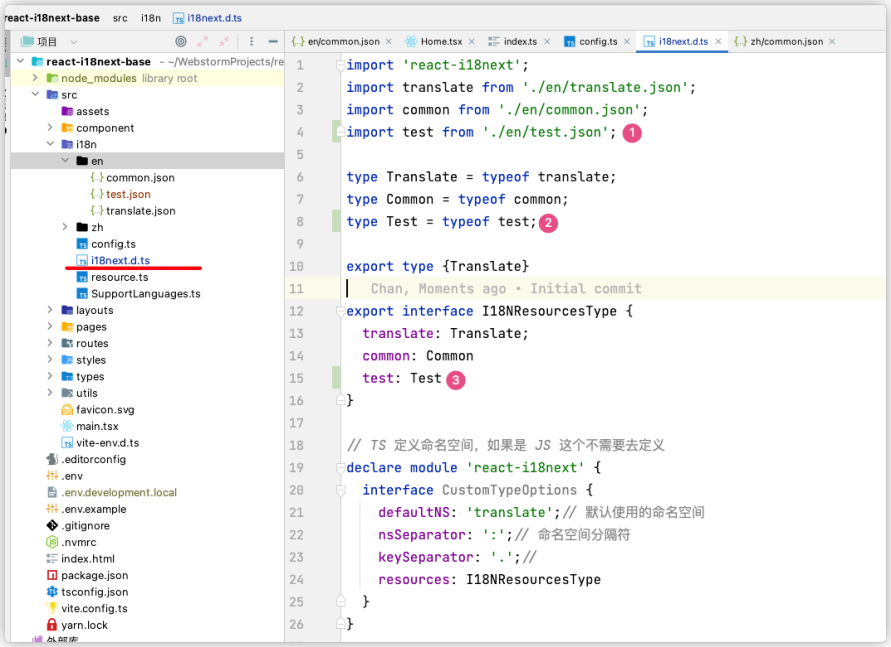
1-3. 新增命名空间

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

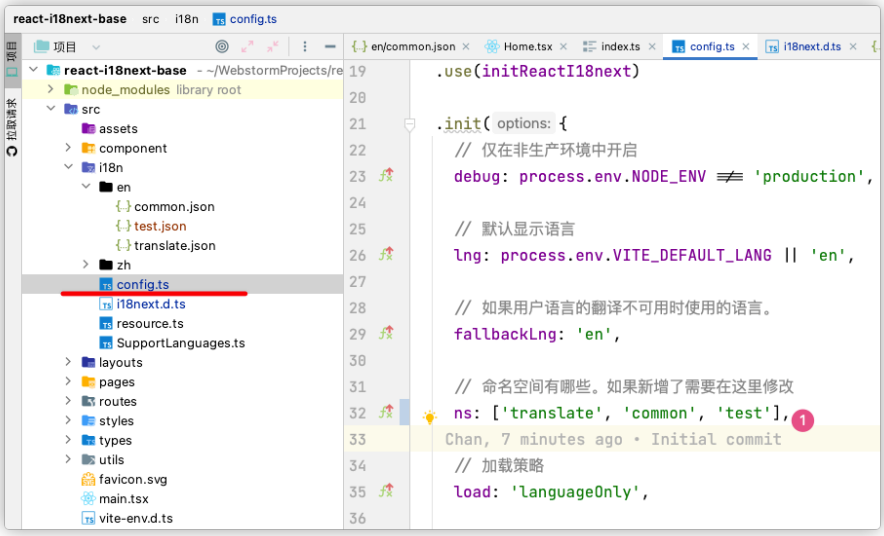


具体流程

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



3. 修改文件 `/src/i18n/config.ts`



2. 函数组件中使用 react-i18next

```tsx
function Case2() {
  // 这种方式可以传入多个
  // const {t} = useTranslation(['translate'])

  const {t} = useTranslation('translate')

  return (
    <div>
      <DisplayText size="large">
        <CodeShow>t('Case2')</CodeShow>:
        {t('Case2')}
      </DisplayText>
    </div>
  )
}
```

3. 类组件中使用 react-i18next

```
import React from "react";
import {WithTranslation} from "react-i18next";

// 注意这里，IProps 接口继承 WithTranslation 并指定命名空间
// 下述是表示指定了使用 translate 命名空间，以获得 IDEA 参数补全
interface IProps extends WithTranslation<'translate'>{
}

export class Home extends React.Component<IProps, any> {
    render() {
        // 看这里
        const {t} = this.props

        // 使用
        return (<h1>{t('Welcome')}</h1>)
    }
}

```

4. `Trans` 组件的使用

```
/**
 * 组件混合翻译
 * @constructor
 */
function Case5() {
  return (
    <Trans ns="translate" i18nKey="Case5">
      案例5：使用 <CodeShow>Trans</CodeShow> Component，了解更多
      <Link external url="https://react.i18next.com/latest/trans-component">点击这里</Link>
    </Trans>
  )
}
```

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

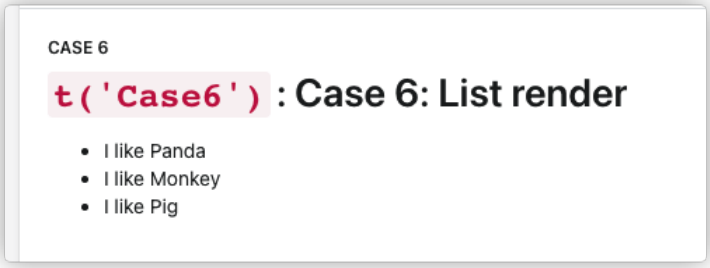
语言包：

```
{
  "AnimalList": [
    "I like Panda",
    "I like Monkey",
    "I like Pig"
  ]
}
```

代码:

```
/**
 * 列表渲染
 * @constructor
 */
function Case6() {
  const {t} = useTranslation('translate')

  return (
    <div>
      <ul>
        {t('AnimalList', {returnObjects: true}).map((item, index) => {
          return <li key={index}>{item}</li>
        })}
      </ul>
    </div>
  )
}
```



7. 单复数

语言包:

```
{
  "Case7": {
    "apple": "I have {{count}} apple",
    "apple_other": "I have <b>{{count}}</b> apples"
  }
}
```

代码:

```jsx
function Case7() {
  return (
    <ul>
      <li>
        <Trans ns="translate" i18nKey="Case7.apple" count={-2}>
          I have {{count: -2}} apple
        </Trans>
      </li>

      <li>
        <Trans ns="translate" i18nKey="Case7.apple" count={-1}>
          I have {{count: -1}} apple
        </Trans>
      </li>

      <li>
        <Trans ns="translate" i18nKey="Case7.apple" count={0}>
          I have {{count: 0}} apple
        </Trans>
      </li>

      <li>
        <Trans ns="translate" i18nKey="Case7.apple" count={1}>
          I have {{count: 1}} apple
        </Trans>
      </li>

      <li>
```

运行结果：

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

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

/**
 * 基础使用
 * @constructor
 */
export function TestUtil_1() {
  return i18next.t('common:Save')
}

/**
 * 返回 object 内容
 * @constructor
 */
export function TestUtil_2() {
  return i18next.t('translate:AnimalList', {returnObjects: true})
}

/**
 * 使用变量
 *
 * @param count
 * @constructor
 */
export function TestUtil_3(count: number = 1) {
  return i18next.t('common:Case_N', {replace: {num: count}})
}

```

JSX 代码:

```jsx
function Case8() {
  console.log('看看是不是对象', TestUtil_2())

  return (<div>
    <DisplayText>非组件上使用 react-i18next</DisplayText>
    <ul>
      <li><CodeShow>TestUtil_1</CodeShow>: {TestUtil_1()}</li>
      <li><CodeShow>TestUtil_2</CodeShow>: {TestUtil_2()}</li>
      <li><CodeShow>TestUtil_3</CodeShow>: {TestUtil_3()}</li>
    </ul>
  </div>)
}
```

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

```
import React from "react";
import {LanguageUtils} from "@/utils/languages";
import {IListenerNameEnum} from "@/utils/languages.d";

export default class extends React.Component<any, any> {
  state = {
    load_time: 0
  }

  // 组件挂载，添加监听器
  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
      this.setState({
        load_time: new Date().getTime
      })
    })
  }

  // 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！！！！！
  componentWillUnmount() {
    LanguageUtils.removeListener(IListenerNameEnum.FirstListener)
  }


```