## 安装

npm install vuex

src目录下新建store目录用于存放store文件，新建main.js作为主store文件。以下是store模板

import Vue from 'vue'

import Vuex from 'vuex'

Vue.use(Vuex);

export default new Vuex.Store({

state:{...},

mutations:{...},

actions:{...},

getters:{...},

modules:{}

})

## **项目mian.js引入store**

import Vue from 'vue';

import App from './App.vue';

**import vuex from 'vuex';**

**import store from './store/main.js'**

Vue.use(vuex)

new Vue({

**store,**

render: h => h(App),

}).$mount('#app')

## **使用state中数据**

### 直接使用（不推荐）

<template>

<div class="hello">

**{{$store.state.loginInfo}}** <!--直接使用-->

</div>

</template>

<script>

export default { name: 'hello'}

</script>

<style scoped></style>

### 调用mapState工具，使用store中数据

<template>

<div class="hello">

hello,{{loginInfo}}

</div>

</template>

<script>

**import { mapState } from 'vuex' //调用工具**

export default {

name: 'hello',

computed:{

**...mapState(['loginInfo'])**//使用工具引出store中数据

}

}

</script>

<style scoped></style>

## **修改state中数据：mutation**

Vuex不推荐直接修改store中的数据，而是通过mutation中自定义的方法进行修改。

【注意】

1. Mutation中不能写异步，延时操作。（异步，延时由store中的action进行）
2. 定义的必需参数state

### 定义

mutations:{

changeLoginInfo(**state**,newObj){

**state**.loginInfo = newObj;

}

},

### 直接调用（不推荐）

//第1个参数是方法名，第二个以后是参数

this.$store.commit('changeLoginInfo',this.loginInfo\_copy);

### 调用mapMutations工具，修改store中的数据

<template>

<div class="hello">

<input type="text" v-model="loginInfo\_copy.name" @keypress.enter="changeInfo"/>

</div>

</template>

<script>

**import { mapMutations } from 'vuex' //引入工具**

export default {

name: 'hello',

components:{ },

beforeMount(){

const keys = Object.keys(this.loginInfo);

keys.forEach( item => this.loginInfo\_copy[item] = this.loginInfo[item]);

},data(){

return{

loginInfo\_copy:{name:''}

}

},methods:{

**...mapMutations(['changeLoginInfo']),//利用工具将方法引入**

changeInfo(){

**this.changeLoginInfo(this.loginInfo\_copy)//执行方法**

}

}

}

</script>

<style scoped>

</style>

## **异步操作state中数据：actions**

actions运行异步操作store中数据，但需通过调用mutation执行。必需参数context

### 定义

actions:{

changeLoginInfoAsync(**context**,newObj){

setTimeout(() => {

**context**.**commit**('changeLoginInfo',newObj);

}, 1000\*2);

}

}

### 直接调用（不推荐）

//第1个参数是方法名，第二个以后是参数

this.$store.dispatch('changeLoginInfoAsync',this.loginInfo\_copy);

### 调用mapActions工具，修改store中的数据

<template>

<div class="hello">

hello

<input type="text" v-model="loginInfo\_copy.name" @keypress.enter="change"/>

</div>

</template>

<script>

import { mapState,mapMutations,**mapActions** } from 'vuex'**//引入工具**

export default {

name: 'hello',

computed:{

...mapState(['loginInfo'])

},

beforeMount(){

const keys = Object.keys(this.loginInfo);

keys.forEach( item => this.loginInfo\_copy[item] = this.loginInfo[item]);

},data(){

return{

loginInfo\_copy:{name:''}

}

},methods:{

...mapMutations(['changeLoginInfo']),

**...mapActions(['changeLoginInfoAsync']),//利用工具引入对应方法**

changeInfo(){

this.changeLoginInfo(this.loginInfo\_copy)

},

change(){

**this.changeLoginInfoAsync(this.loginInfo\_copy);//调用方法**

}

}

}

</script>

<style scoped>

</style>

## **修饰state中数据：getter**

【注意】Getter不会修改store中的数据，只会包装store中的数据，可以将getter看作是computed。State是必需参数。

### 定义

getters:{

getName(**state**){

return state.loginInfo.name;

}

},

### 直接调用

**this.name = this.$store.getters.*getName//这里使用调用属性方式调用getter***

### 调用mapGetters工具，调用store中包装后的数据

<template>

<div class="hello2">

{{**getName**}}

</div>

</template>

<script>

**import {mapGetters} from 'vuex';**

export default {

name: 'hello2',

computed:{

**...mapGetters(['getName'])**

}

}

</script>

<style scoped>

</style>

## **模块化——modules**

使用模块化将state拆分管理，虽说vuex结构可以根据自己需要定义，但尽量遵守如下结构：

目录结构

src

|—store

|—index.js 汇总所有的modules

|—modules

|—modulesA

|—index.js

|—modulesB

|—index.js

### 主Index.js

import Vue from 'vue'

import Vuex from 'vuex'

//引入模块

import car from './modules/car/index';

import per from './modules/per/index';

Vue.use(Vuex);

export default new Vuex.Store({

state:{

},getters:{

},mutations:{

},actions:{

},modules:{

car,per

}

})

### 定义模块

【注意】命名控件很重要，可避免不同模块的相同方法或store值的错误提示。

export default {

**namespaced:true,**

state:{

name:'car'

},

getters:{

getName(state){

return state.name;

}

}

}

export default {

**namespaced:true,**

state:{

name:'car'

},

getters:{

getName(state){

return state.name;

}

}

}

### 调用store

直接调用

<!--$store.state.命名空间.store名-->

{{$store.state.car.name}}

{{$store.state.per.name}}

利用mapState工具调用

{{pername}},{{carname}}

import {mapState} from 'vuex'

computed:{

...mapState({

pername : state => state.per.name,

carname : state => state.car.name

})

},

### 调用mutation

直接调用

this.$store.commit(

'per/changeName',

this.name

);

利用mapMutations工具调用

import {mapMutations} from 'vuex'

methods:{

...mapMutations({

**changepname:'per/changeName',**

**changecname:'car/changeName'**

}),changeCName2(){//调用

this.changecname(this.name);

},changePName2(){//调用

this.changepname(this.name);

}

}

### 调用actions

直接调用

this.$store.dispatch(

'per/changeNameAsync',

this.name

);

利用mapActions工具调用

import {mapActions} from 'vuex'

methods:{

...mapActions({

**changeNameAsyncP:'per/changeNameAsync',**

**changeNameAsyncC:'car/changeNameAsync'**

}),changeP(){

this.changeNameAsyncP(this.name);

},changeC(){

this.changeNameAsyncC(this.name);

}

}

### 调用getter

直接调用

{{$store.getters['car/getName']}}

利用mapGetter工具调用

import {mapGetters} from 'vuex'

computed:{

...mapGetters({

**getPername:'per/getName',**

**getCarname:'car/getName',**

})

｝

## **插件**

Vuex社区提供不少插件