Vuex进阶

⼀ state

、

* 1. 引⼊vuex 以后，我们需要在state中定义变量，类似于vue中的data，通过state来存放状态

import Vue from 'vue' import Vuex from 'vuex' Vue.use(Vuex)

export default new Vuex.Store({ state: { //存放状态

nickname:'Simba', age:20,

gender:'男'

},

mutations: {},

actions: {}, modules: {}

})

注册两个组件分别引⼊到app.vue中

<div id="app">

<vabout> </vabout>

<vhome> </vhome>

</div>

vhome组件内容

<div class="home">{{$store.state.nickname}}</div>

vabout组件内容

<h1>{{$store.state.nickname}}:{{$store.state.age}}</h1>



如图，显示出显示出相应的内容，有了vuex，我们不必在考虑组件之间的传值，直接就可以通过$store来获取不同的数据，但是如果需要vuex中的多个 数据的这时候，这样写就太啰嗦了，我们可以将它定义在computed中

Props ， methods , data 和 computed 的初始化都是在 beforeCreated 和 created 之间完成的。例：

<template>

<div class="home">

{{nickname}}

</div>

</template>

<script>

export default { name: 'home', computed:{

nickname(){

return this.$store.state.nickname

}

}

}

</script>

##### 这样引⼊就⽅便了很多。

* 1. mapState 辅助函数

1.1中的⽅法虽然引⼊的时候⽅便了，但是computed中定义的代码还是很多，⽽这时候vuex⼜给我们提供了更简便的⽅法mapState⽅法

import {mapState} from 'vuex' export default {

name: 'home',

computed: mapState(['nickname','age','gender'])

}

mapState(['nickname','age','gender']) //映射哪些字段就填⼊哪些字段

这⼀句代码就相当于下⾯这些

nickname(){return this.$store.state.nickname} age(){return this.$store.state.age} gender(){return this.$store.state.gender}

##### 记住：⽤mapState等这种辅助函数的时候，前⾯的⽅法名和获取的属性名是⼀致的。如果我们需要⾃定义⼀个计算属性怎么办呢？怎么添加呢？

毕竟现在已经成这样了 computed: mapState(['nickname','age','gender']) 这时候我们就需要es6中的展开运算符：...

computed: { //computed是不能传参数的value(){

return this.val/7

},

...mapState(['nickname','age','gender'])

}

### ⼆ getters

、

2.1 getters相当于vue中的计算属性，通过getters进⼀步处理，得到我们想要的值，⽽且允许传参，第⼀个参数就是state

import Vue from 'vue' import Vuex from 'vuex' Vue.use(Vuex)

export default new Vuex.Store({ state: { //存放状态

nickname:'Simba', firstname:'张',

lastname:'三丰',

age:20, gender:'男', money:1000

},

getters:{ realname(state){

return state.firstname+state.lastname

},

money\_us(state){

return (state.money/7).toFixed(2)

}

},

mutations: {},

actions: {}, modules: {}

})

vue部分

computed: { //computed是不能传参数的valued(){

return this.value/7

},

...mapGetters(['realname','money\_us'])

}

### 三 Mutation

、

* 1. 我们代码中定义的时候需要些mutations，它类似于vue中的methods，

mutations需要通过commit来调⽤其⾥⾯的⽅法，它也可以传⼊参数，第⼀个参数是state，第⼆个参数是载荷（payLoad），也就是额外的参数 代码如下

mutations: { //类似于methods addAge(state,payLoad){

state.age+=payLoad.number

}

}

#### template部分

<div class="home">

<div><button @click="test">测试</button></div>

</div>

js部分

methods:{ test(){

this.$store.commit('addAge',{ number:5

})

}

}

##### 调⽤的时候第⼆个参数最好写成对象形式，这样我们就可以传递更多信息。

但是，这样写还是会遇到同样的问题，就是如果需要操作多个数据，就会变的麻烦，这时候我们就需要mapMutations，通过它将⽅法映射过来

* 1. mapMutations

跟mapState、mapGetters⼀样

methods:{

...mapMutations(['addAge'])

}

mapMutations(['addAge'])这⼀句就相当于下⾯的代码

addAge(payLoad){ this.$store.commit('addAge',payLoad)

}

##### 参数我们可以在调⽤这个⽅法的时候写⼊

<button @click="addAge({number:5})">测试</button>

这时候⼀些杠精就要说了，我为什么要绕⼀圈，从mutations⾥⾯去改state呢？我能不能直接改state呢？

##### ⽐如这样：

addAge(){ this.$store.state.age +=5;

}

实际看结果也可以，那我为什么从mutations⾥⾯中转⼀下呢？ 原因如下：

① 在mutations中不仅仅能做赋值操作

② 作者在mutations中做了类似埋点操作，如果从mutations中操作的话， 能被检测到，可以更⽅便⽤调试⼯具调试，调试⼯具可以检测到实时变化，

⽽直接改变state中的属性，则⽆法实时监测

注意：mutations只能写同步⽅法，不能写异步，⽐如axios、setTimeout等，这些都不能写，mutations的主要作⽤就是为了修改state的

原因类似：如果在mutations中写异步，也能够调成功，但是由于是异步的，不能被调试⼯具追踪到，所有不推荐这样写，不利于调试

定

* 1. 使⽤常量替代Mutation事件类型

把原本的⽅法名称由字符串转变成常量代码如下：

import Vue from 'vue' import Vuex from 'vuex'

export const ADD\_AGE ='addAge' Vue.use(Vuex)

export default new Vuex.Store({ state: { //存放状态

nickname:'Simba', firstname:'张',

lastname:'三丰',

age:20, gender:'男', money:1000

},

getters:{ //类似于 computed

realname:state =>state.firstname+state.lastname, money\_us(state){

return (state.money/7).toFixed(2)

}

},

mutations: { //类似于methods [ADD\_AGE](state,payLoad){

state.age+=payLoad.number

}

},

actions: { }, modules: {}

})

将addAge⽅法名字定义为⼀个常量，当调⽤的时候直接引⼊

import {ADD\_AGE} from '../store' import {mapMutations} from 'vuex' export default {

methods:{

...mapMutations([ADD\_AGE])

}

}

这样写的好处：

① 不容易写错，字符串容易写错，⽽且字符串写错以后不会报错位置，⽽⽤常量替代，如果写错，eslint可以提示错误位置

⽤常量替代mutations的时候我我们可以新建⼀个⽂件（mutation\_type.js）专⻔存储这些常量mutation\_type.js部分

export default { ADD\_AGE: ‘addAge’

}

然后再store/index.js中引⼊

import MUTATION\_TYPES from ‘./mutation\_type’（先引⼊）

export let MUTATION\_TYPE=MUTATION\_TYPES （再导出）

这个地⽅有⼀个坑，不要将引⼊和导出合并成⼀⾏代码：⽐如这样

export { foo, bar } from 'my\_module';

// 可以简单理解为

import { foo, bar } from 'my\_module'; export { foo, bar };

需要注意的是，两者并不⼀样，写成⼀⾏以后，foo和bar实际上并没有被导⼊当前模块，只是相当于对外转发了这两个接⼝，导致当前模块不能直接使

⽤foo和bar。

vue部分

import {MUTATION\_TYPE} from '../store' methods:{

...mapMutations([MUTATION\_TYPE.ADD\_AGE])

}

总结⼀下：

① 依赖state得到新的数据，⽤getters（跟computed⼀样，只读）

② 修改state的属性值，就⽤mutations（同步操作）

四 actions

、

#### 4.1 action类似于mutation

区别：action可以提交mutation

action也不要直接去操作state，⽽是去操作mutation

action包含异步操作，类似于axios请求，可以都放在action中写action中的⽅法默认的就是异步，并且返回promise

代码如下store部分

actions: { getUserInfo(){

return { nickname:'Simba', age:20

}

}

}

在actions中定义⼀个⽅法：getUserInfo，并且返回⼀个对象vue部分

created(){

var res = this.getUserInfo() console.log(res)

},

methods:{

...mapActions(['getUserInfo'])

}

在created中调⽤此⽅法，并将结果赋值给res，打印res，结果打印出Promise



这表明，在actions中的⽅法，默认就是异步的，通过then获取数据mapActions(['getUserInfo']) 相当于以下代码

getUserInfo(){

return this.$store.dispatch(‘getUserInfo’)

}

在实际开发当中，state⾥⾯的属性值是空的，当登录以后，再进⾏获取对应的信息。登录以后，需要得到⽤户信息，那如何得到呢？

⾸先进⼊⻚⾯的时候调⽤actions中的getUserInfo⽅法代码如下

vue部分

created(){ this.getUserInfo()}

methods:{ ...mapActions([‘getUserInfo’])}

store部分

⾸先要想得到数据，那就相当于给state赋值，那⾸先想到的就是mutations来操作state，但是请求的接⼝都是axios异步，所以就不能⽤mutations⽽是

⽤actions，通过actions来操作mutations从⽽操作state

export default new Vuex.Store({ state: {

nickname:‘’, age:0, gender:'', money:0

},

mutations: { setUerInfo(state,payLoad){

state.nickname = payLoad.nickname state.age = payLoad.age state.gender = payLoad.gender state.money = payLoad.money

}

},

actions: { //actions没有提供state当参数async getToken({commit}){

var res = await axios.get('/token接⼝') commit('setToken',res)

},

async getUserInfo(context){

//context可以理解为它是整个Store的对象.类似于this.$store， 他⾥⾯包含了state，getter，mutations，actions

const res = await axios.get('/接⼝url') context.commit('setUerInfo',res)

//相当于 this.$store.commit,第⼀个参数是⽅法名，第⼆个参数是要传⼊的数据context.dispatch('getToken')

//actions也可以调⽤⾃⼰的其他⽅法

},

}

})

运⾏过程，调⽤getUserInfo⽅法以后，进⼊actions，然后通过commit调⽤setUserInfo，将res（⽤户信息）作为参数传⼊传⼊进去，并将相对应的属 性值赋值给state，完成这⼀过程的操作。

getUserInfo的参数也可以⽤解构，这样更⽅便

async getUserInfo({commit,dispatch}){ const res = await axios.get('/接⼝url') commit('setUerInfo',res) dispatch('getToken')

}