！阅读本文档。你需要对vuex有一定的了解！

如果你的项目比较小。不要采用本框架，因为没有必要

1.本项目基于vue-cli提供思路，在其他项目中需要用，直接复制store文件夹然后在App.js中引入并在Vue中注入即可。项目的构建思路和react的redux封装的dva脚手架思路相近。

1. 大体思路：

modules:项目以vue路由页面为单位。因为每一次路由改变，store中state所有值都会复原。

所以modules以路由组件为单位而不是以单个组件为单位。一个路由组件包括其所有子 组件的状态和数据都应该存入到一个当前module中

actions:

将所有异步操作包括数据请求都放入到actions层中，actions由页面事件或vue路由钩 子中通过dispatch函数触发。数据请求的结果通过commit multations来反映到state 状态中去。

multations:multations是唯一能改变state的方式，只能执行同步操作，在页面中由事件或者 路由钩子触发。

getters:相当于state的过滤属性。没多少可说的。

state:数据仓库，你可以将和页面所有有关的数据存储到其中，也可以将一些不重要的东西 （比如控制组件状态的数据）放到组件中，看个人喜好。

3.注意：需要将vuex和数据缓存区分开来。这两部分业务是互相独立的，不要混淆。

1. 配置vuex和及其项目结构

（1）搭建你的vue项目结构。此点略过

（2）npm install vuex --save

main.js: 引入的Vuex V要大写

import store from './store/index'

new Vue({//注入store

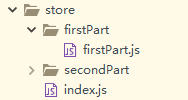
el: '#app',

store,

render: h => h(App)

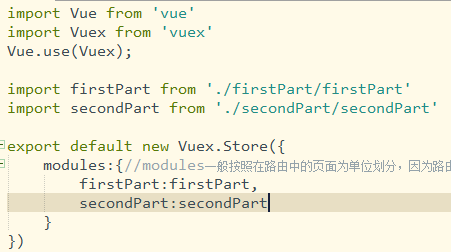
})

1. 项目下建立store文件夹,可以说除了组件内需要映射state\getters或者触发actions和multations，你现在可以把所有精力都放到store及其以下的文件夹中来。



store文件夹如上

Index.js：相当于vuex的入口文件



固定写法，导出store后只需要在new Vue中注入store，你就可以在组件中使用vuex

firstPart....等就是你的mudule,匹配到相应路由组件完成功能

1. modulse:(这里以firstPart为例子)，关于namespace设置为true和false会大大影响到vuex参数的写法和用法，本项目基于大型项目的考虑，将其设置为true。不考虑为false的情况。

import {serviceOne} from '../../services/firstPartService';

//这里存放所有异步请求数据的方法，统一管理，也就是说，actions中的所有异步请求，都//放到其中

//下面会遇到的参数名

参数名：state,getters:当前module state和getters,

rootState、rootGetters:根state和根getters

Dispatch,commit:触发其他actions和multations的两个函数需要注入

params/payload:各种multations和actions函数之间触发需要传递的参数

const firstPart = {

namespaced:true,

state:{//将当前组件所有状态和数据存入其中通过mapState映射到组件中去

name:""

.

.

.

},

getters:{//就是state的计算属性，通过mapGetter映射到组件中去

gettersThree(state,getters,rootState,rootGetters){

return ''

}

},

mutations:{

//在页面中用this.$store.commit('firstPart/mutationsOne',param)来触发（此时 //mutations全局可访问，所以所有mutations命名尽量避免重名）

mutationsOne(state,payload){

console.log(state);//当前命名空间state

console.log(payload);//组件传过来的参数

}

},

actions:{

// 3.本项目使用async函数(这一条需要babrl转码generator函数才行)

//1.npm install --save-dev babel-polyfill

//2.webpack.config.js:var babelpolyfill = require("babel-polyfill");

//3.main.js import "babel-polyfill";

// this.$store.dispatch('firstPart/actionsFour');

}

}

export default firstPart

1. 组件内使用vuex:

(1) state的获取方式: ...mapsatate()函数,

computed:{

...是ES6规范，需要使用按以下方式

.babelrc

"presets": [

"es2015",

"stage-3",

],

npm install --save-dev babel-cli

npm install --save-dev babel-preset-es2015

npm install --save-dev babel-preset-stage-3

...mapState('firstPart',{//前面相当于命名空间的路径

//前面是计算属性真名，后面是需要从state映射过来的值

nameTwo:state => state.name

})

},

(2)getters的获取

computed: {

...mapGetters('firstPart',{

gettersOne:'gettersThree'

})

}

(3)：mutations和actions的触发方法,这两个方法也可以在store里面使用，引入commit 和dispatch后直接使用

mounted(){

1.mutations

this.$store.commit('firstPart/mutationsOne',{param:'页面commit传递的值'});

2.actions

this.$store.dispatch('firstPart/actionsOne',{params:'1'}).then(()=>{})

}