1. Vuex是什么

Vue.js应用程序开发的状态管理模式

集中式存储管理应用的所有组件的状态

1. 什么情况下使用

中大型的单页面

1. 开始

每一个vuex应用的核心就是store（仓库）；store是一个容器，包含着应用的大部分状态。

Vuex和单纯全局对象不同之处：

1、Vuex的存储状态是响应式的。（store变，组件的引用也会更新）

2、不能直接改变store的值。

改变store中状态的唯一方法：

显示的提交（commit）mutation

1. State介绍

Vuex使用单一状态树，用一个对象就包含了全部的应用层级状态。 唯一数据源

每个应用仅仅包含一个store实例

1. 在 Vue 组件中展示状态（从store中读取）

在计算属性中返回某个状态

// 创建一个Counter组件



（每当store.state.count变化的时候，都会重新求取计算属性，并且触发更新相关联的DOM）

这种模式导致组件依赖全局状态单例，在模块化的构建系统中，每个需要state的组件都需导入

1. Vuex的store选项

将状态从根组件注入到每一个子组件中（需调用Vue.use(Vuex)）

子组件通过this.$store访问



1. mapState辅助函数
2. 使用场景 一个组件需要获取多个状态时
3. 作用 生成计算属性
4. 引入 import {mapState} from ‘vuex’
5. computed: mapState( {

count： state => state.count

} )

1. 映射的计算属性的名称与 state 的子节点名称相同时,可以给mapState传一个字符串数组



1. 对象展开运算符

mapState函数返回一个对象

将mapState对象与局部计算属性混合

...mapState({

})

1. Mutation（同步事物）

更改 Vuex 的 store 中的状态的唯一方法是提交 mutation

每个 mutation 都有一个字符串的 事件类型 (type) 和 一个 回调函数 (handler)

回调函数就是我们实际进行状态更改的地方

state 作为回调函数第一个参数



要唤醒一个 mutation handler，你需要以相应的 type 调用 store.commit 方法

store.commit('increment')

使用常量替代 Mutation 事件类型

[SOME\_MUTATION] (state) { // mutate state }

Mutation 必须是同步函数

在组件中提交 Mutation

this.$store.commit('xxx')

1. Action（异步事物）

action类似于mutation

Action与mutation的不同：

|  |  |
| --- | --- |
| Action | mutation |
| 提交的是mutation | 直接变更状态 |
| 可以包含任意异步操作 | 必须是同步执行 |

Action函数接受一个context对象

通过context.commit提交一个mutation

通过context.state和context.getters来获取

Action 通过 store.dispatch 方法触发：

store.dispatch('increment')

在组件中使用 this.$store.dispatch('xxx') 分发 action

五、最简单的Store

创建过程仅需要提供一个初始state对象和一些mutation

const store = new Vuex.Store({

state :{

count: 0

},

mutation: {

Increment(state){

state.count++

}

} // store.commit方法出发状态改变

}) // store.state 获取对象

store.commit( ‘Increment’ );

console.log( store.state.count ); // 1

由于store中的状态是响应式的，在组件中调用store中的状态：计算属性中返回即可

出发变化也仅是在组件methods中提交mutation

import {mapState} from 'vuex';

computed: {  
 ...mapState([  
 'user'  
 ])  
 }