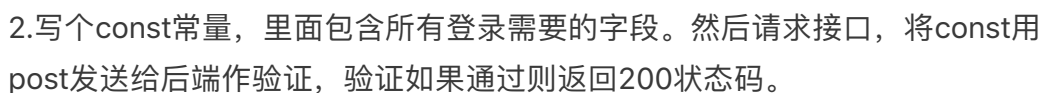


1、先用postman测试登录接口，将需要的参数传进去，看可否拿到access_token,如果拿到则登录成功。



3.返回200状态码之后，将接口返回的`token_type`和`access_token`存储在浏览器的`localStorage`里面。

4.然后做跳转到首页。

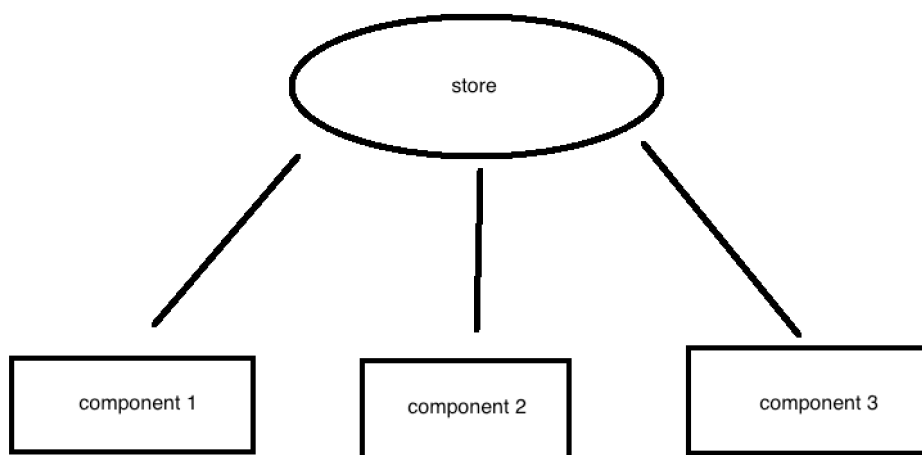
(注册后的用户会有自己独一无二的token, 格式:

Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJ



- 1.主要应用于Vue.js中管理数据状态的一个库
- 2.通过创建一个集中的数据存储，供程序中的组件们访问。

下图可以理解为“单一的数据源”



建立好之后所有的组件都可以从这个store里面读取数据。并对数据进行处理。

vuex小练习思路：

- 1.先跑一个新的带有vuex的vue项目
- 2.建立两个内容为列表的子组件
- 3.将在父组件中将子组件们绑定数组

```
<teacher-list-one v-bind:teachers = 'teachers'><
```

- 4.去子组件中去用props接收值

```
export default {  
  props:['teachers'],  
}
```

5.数据可以成功显示后，就开始改用computed获取store的数据

```
computed: {  
  teachers() {  
    return this.$store.state.teachers  
  },  
}
```

循环数据的地方不需要做更改，现在就可以读到store里面state的值了。

6. (现在增加需求，假设某场景需要每个组件都需要改变state数据的值)

组件一：

```
computed: {  
  teachers() {  
    return this.$store.state.teachers  
  },  
}
```

组件二：（在此组件中让数据name改名，且num变成两倍）

```
export default {  
  computed: {  
    teachers() {  
      return this.$store.state.teachers  
    },  
    changeTeachers() {  
      let changeTeachers = this.$store.state.teachers.map(teacher => {  
        return {  
          name: 'new' + teacher.name,  
          num: teacher.num * 2  
        }  
      })  
      return changeTeachers  
    }  
  },  
}
```

写完这些你会发现两个子组件中对同一数据做出了不同的反应。反思，如果我同时需要有一百个组件同时更改数据变两倍，那么我需要重复使用changeTeachers代码一百遍...

所以此刻我们开始使用vuex的getters属性。

将上述changeTeachers方法整个剪切到store里面getters里面去，再将state作为参数传进去。

```

    getters:{
      changeTeachers(state){
        let changeTeachers = state.teachers.map(teacher =>{
          return {
            name:'new' + teacher.name,|
            num:teacher.num * 2
          }
        })
        return changeTeachers
      }
    }
  }

```

而子组件代码中，则可以优化为：

```

    changeTeachers(){
      return this.$store.getters.changeTeachers
    }

```

7.mutations帮你触发方法去更改store里面的数据（唯一办法）

（此时的场景需求是，点击按钮，能降低/增加子组件的num数量）

```

    methods:{
      reduceClass(){
        this.$store.state.teachers.map(teacher =>{
          teacher.num -= 1
        })
      },
      increaseClass(){
        this.$store.state.teachers.map(teacher =>{
          teacher.num += 1
        })
      }
    }
  }

```

这种写法虽然实现变更数据，但在vue插件中不好调试，看不到vuex数据的变更

所以我们去store里面去写一个mutation

```

    mutations: {
      reduceClass(state){
        state.teachers.map(teacher =>{
          teacher.num -= 1
        })
      }
    },

```

然后将子组件代码改变成

```

methods:{
  reduceClass(){
    // this.$store.state.teachers.map(teacher =>{
    //   teacher.num -= 1
    // })
    this.$store.commit('reduceClass') //触发mutations里面的方法
  },
  increaseClass(){
    this.$store.state.teachers.map(teacher =>{

```

8.actions提交的是mutation，而非直接变更状态。并且可以包括任意的异步操作。

(做一个3秒后执行变化的小测试)

先在mutation里面加上延迟执行，会发现vue插件里面点击会立刻触发方法，但数据会延迟变更。

此时就把方法写到actions里面

```

actions: {
  reduceClass(context,payload){
    setTimeout(function () {
      context.commit("reduceClass",payload)
    },2000)
  }
}

```

这样之后，去子组件触发actions，

```

this.$store.dispatch("reduceClass")

```

这样就会看到vue插件里面触发和变更同步进行。

有时候需要变更的数量不定，需要我们传递参数的时候，则click方法里的小括号里面写上要传的数字


```

,
methods:{
  reduceClass(count){
    // this.$store.state.teachers.map(teacher =>{
    //   teacher.num -= 1
    // })
    // this.$store.commit('reduceClass') //触发mutat
    this.$store.dispatch("reduceClass",count)
  },
  increaseClass(){

```

然后store里面加上载荷payload：

```

mutations: {
  reduceClass(state, payload){
    // setTimeout(function () {
    state.teachers.map(teacher =>{
      teacher.num -= payload
    })
    // }, 3000)
  }
},
actions: {
  reduceClass(context, payload){
    // 
    setTimeout(function () {
      context.commit("reduceClass", payload)
    }, 2000)
  }
}
}

```

这样再点击减少课程按钮，会发现每次减少的，是你传进去的数字。

9. 上述代码理解之后，可进一步优化代码

```

import {mapGetters, mapActions} from 'vuex'
export default {
  computed: {
    teachers() {
      return this.$store.state.teachers
    },
    ...mapGetters([
      'changeTeachers'
    ])
  },
  data() {
    return {}
  },
  methods: {
    ...mapActions([
      "reduceClass"
    ]),
    increaseClass() {
      this.$store.state.teachers.map(teacher => {
        teacher.num += 1
      })
    }
  }
}

```

