

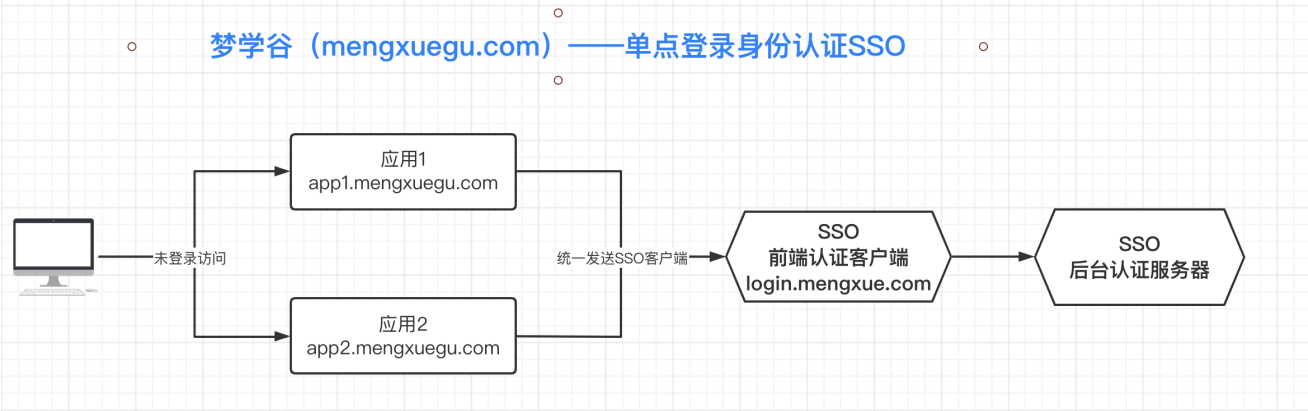
第一章 单点登录实现流程

1.1 背景

在企业发展初期，企业使用的系统很少，通常一个或者两个，每个系统都有自己的登录模块，运营人员每天用自己的账号登录，很方便。

但随着企业的发展，用到的系统随之增多，运营人员在操作不同的系统时，需要多次登录，而且每个系统的账号都不一样，这对于运营人员来说，很不方便。于是，就想到是不是可以在一个系统登录，其他系统就不用登录了呢？这就是单点登录要解决的问题。

单点登录英文全称Single Sign On，简称就是SSO。它的解释是：**在多个应用系统中，只需要登录一次，就可以访问其他相互信任的应用系统。**



图中有4个系统，分别是应用1、应用2 和 SSO。应用1、应用2、应用3没有登录模块，而 SSO 只有登录模块，没有其他的业务模块，当应用1、应用2 需要登录时，将跳到SSO系统，SSO系统完成登录，其他的应用系统也就随之登录了。这完全符合我们对单点登录（SSO）的定义。

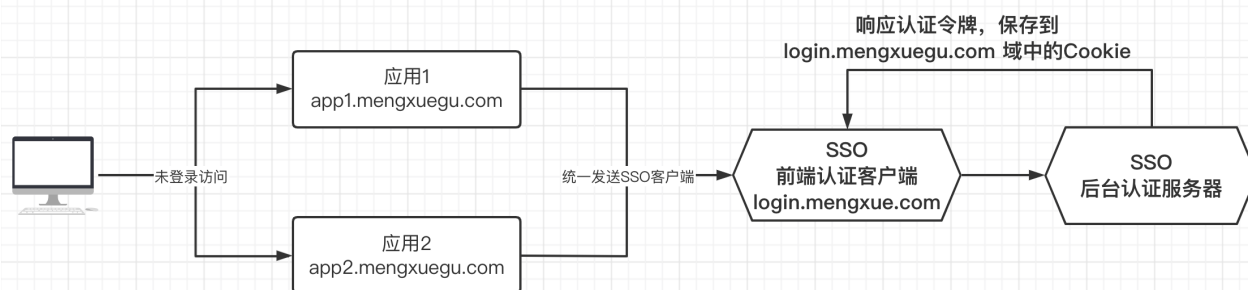
1.2 基于同域下 Cookie 实现 SSO

同一个企业的各种系统，一般情况下只有一个域名，通过二级域名区分不同的系统。比如我们有个域名叫做：mengxuegu.com，同时有两个业务系统分别为：app1.mengxuegu.com 和 app2.mengxuegu.com 。

我们要做单点登录（SSO），就需要一个登录系统，叫做：login.mengxuegu.com。

我们目的是只要在 login.mengxuegu.com 登录，app1.mengxuegu.com 和 app2.mengxuegu.com 也就登录了。

梦学谷 (mengxuegu.com) ——单点登录身份认证SSO



通过上面的登陆认证机制, 我们可以知道, 在 login.mengxuegu.com 中登录了, 其实是在 login.mengxuegu.com 的服务端认证中心记录了登录状态, 并响应了登录状态 (令牌) 给浏览器, 浏览器 (Browser) 将登录状态 (令牌) 写入到 login.mengxuegu.com 域下的 Cookie 中。

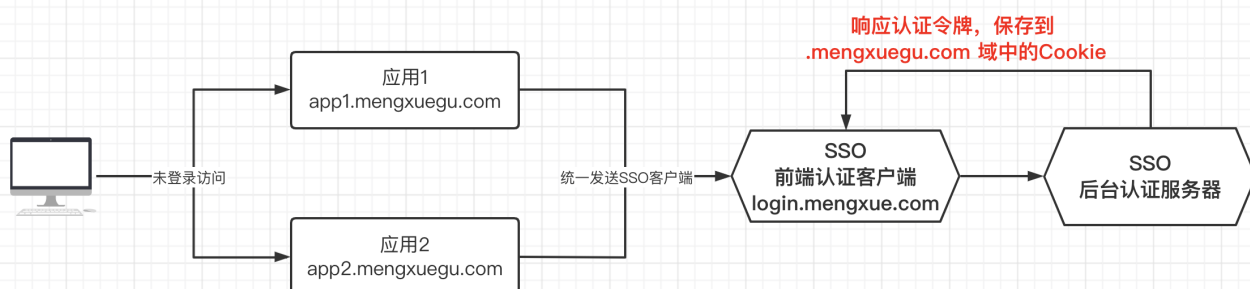
那么我们怎么才能让 app1.mengxuegu.com 和 app2.mengxuegu.com 登录呢? 这里下面问题:

- Cookie 是不能跨域的, 我们 Cookie 的 domain 值是 login.mengxuegu.com, 而在 app1.mengxuegu.com 和 app2.mengxuegu.com 发送请求是获取不到 domain 值是 login.mengxuegu.com 的 Cookie, 从而请求时带上访问令牌的。

针对这一个问题, sso登录以后, 可以将 Cookie 的域设置为顶域, 即 `.mengxuegu.com`, 这样所有子域的系统都可以访问到顶域的Cookie。这样 Cookie 的问题解决了。

我们在设置Cookie时, 只能设置顶域和自己的域, 不能设置其他的域。比如: 我们不能在自己的系统中给 baidu.com的域设置Cookie。

梦学谷 (mengxuegu.com) ——单点登录身份认证SSO



第二章 基于 Vue-CLI 脚手架创建项目

2.1 安装 node.js 和 npm

1. 下载 <http://nodejs.cn/download/>

位于: 04-配套软件\前端软件\node-v14.2.0-x64.msi

闪电PDF编辑器

<https://www.callmysoft.com/>

2. 安装

查看 npm 版本

```
npm -v
```

在 vscode 终端执行 npm 报错：无法将“npm”项识别为 cmdlet、函数、脚本文件或可运行程序的名称
配置 node.js 安装目录到环境变量中, 重启电脑。

3. 配置 npm 淘宝镜像

- 命令

```
npm config set registry https://registry.npm.taobao.org
```

- 验证命令

```
npm config get registry
```

如果返回<https://registry.npm.taobao.org>, 说明镜像配置成功。

2.2 安装 Vue-CLI 脚手架 (不需要安装)

1. 设置全局安装模块保存目录

```
npm config set prefix 'D:\02-devInstall\npm'
```

2. 查看全局保存目录

```
npm root -g
```

3. 全局安装 vue-cli 脚手架

```
$ npm install -g @vue/cli
```

如用的是 MAC 系统, 可能报错: permission denied, access '/usr/local/lib/node_modules'

```
npm ERR! Error: EACCES: permission denied, access '/usr/local/lib/node_modules'
npm ERR! [Error: EACCES: permission denied, access '/usr/local/lib/node_modules'] {
npm ERR!   stack: "Error: EACCES: permission denied, access '/usr/local/lib/node_modules'",
npm ERR!   errno: -13,
npm ERR!   code: 'EACCES',
npm ERR!   syscall: 'access',
npm ERR!   path: '/usr/local/lib/node_modules'
npm ERR! }
```

闪电PDF编辑器

原因: 执行命令行命令时没有获得管理员权限, 输入电脑的管理员密码 (开机密码) 解决: 命令前加上

sudo

```
$ sudo npm install -g @vue/cli
```

4. 安装成功后, 命令行使用 vue 命令, 查看版本号

```
vue -V
```

如果执行上面报如下错, 则要配置环境变量

```
C:\Users\Administrator>vue
'vue' 不是内部或外部命令, 也不是可运行的程
或批处理文件。
```

配置 `vue.cmd` 所在的 `D:\02-devInstall\npm` 目录到环境变量的 path 变量值后面

此电脑 > 安装文件 (D:) > 02-devInstall > npm

名称	类型
node_modules	文件夹
vue	文件
vue.cmd	Windows 命

5. 在 vscode 终端执行 vue 报如下错: 无法将“vue”项识别为 cmdlet、函数、脚本文件或可运行程序的名称

```
vue : 无法将“vue”项识别为 cmdlet、函数、脚本文件或可运行程序的名称。请检查名称的拼写，如果包括路径，请确保路径正确，然后再试一次。
所在位置 行:1 字符: 1
+ vue -V
+ ~~~~
+ CategoryInfo          : ObjectNotFound: (vue:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException
```

解决方式: 配置环境变量后, 要重启电脑

6. 重启后, 如果报错: vue: 无法加载文件 D:\02-devInstall\npm\vue.ps1, 因为在此系统上禁止运行脚本

解决方式: 把 D:\02-devInstall\npm 目录下的 vue.ps1 文件删除。

2.3 导入单点登录项目模板

课程资料中提供了创建好的模板项目, 位于: 03-配套资料/前端单点登录资料/mengxuegu-auth-center.zip

此模板安装好了 axios, 和标题 icon 更新了, 还在 vue.config.js 配置了代理转化

直接解压即可, 然后运行以下命令运行:

闪电PDF编辑器

<https://www.callmysoft.com/>

```
# 进入解压后的项目目录
cd mengxuegu-auth-center
# 运行项目
npm run dev
```

如启动有报错，将 node_modules 目录删除，然后执行 npm install

启动后访问 <http://localhost:7000/>



Hello，梦学谷——陪你学习，伴你梦想

运行vue项目发现报错，如图所示：‘vue-cli-service’不是内部或外部命令，也不是可运行的程序或批处理文件

```
'vue-cli-service' 不是内部或外部命令，也不是可运行的程序
或批处理文件。
npm ERR! code ELIFECYCLE
npm ERR! errno 1
npm ERR! mengxuegu-auth-center@0.1.0 dev: `vue-cli-service serve`
npm ERR! Exit status 1
npm ERR!
npm ERR! Failed at the mengxuegu-auth-center@0.1.0 dev script.
npm ERR! This is probably not a problem with npm. There is likely additional logging output above.
npm ERR! A complete log of this run can be found in:
```

解决办法：将项目里的“node_modules”文件夹删除（不要在 vscode 里删除，在资源管理器按住 Shift + Delete 彻底删除），然后重新运行 npm install

重新下载完成之后，执行以下命令：

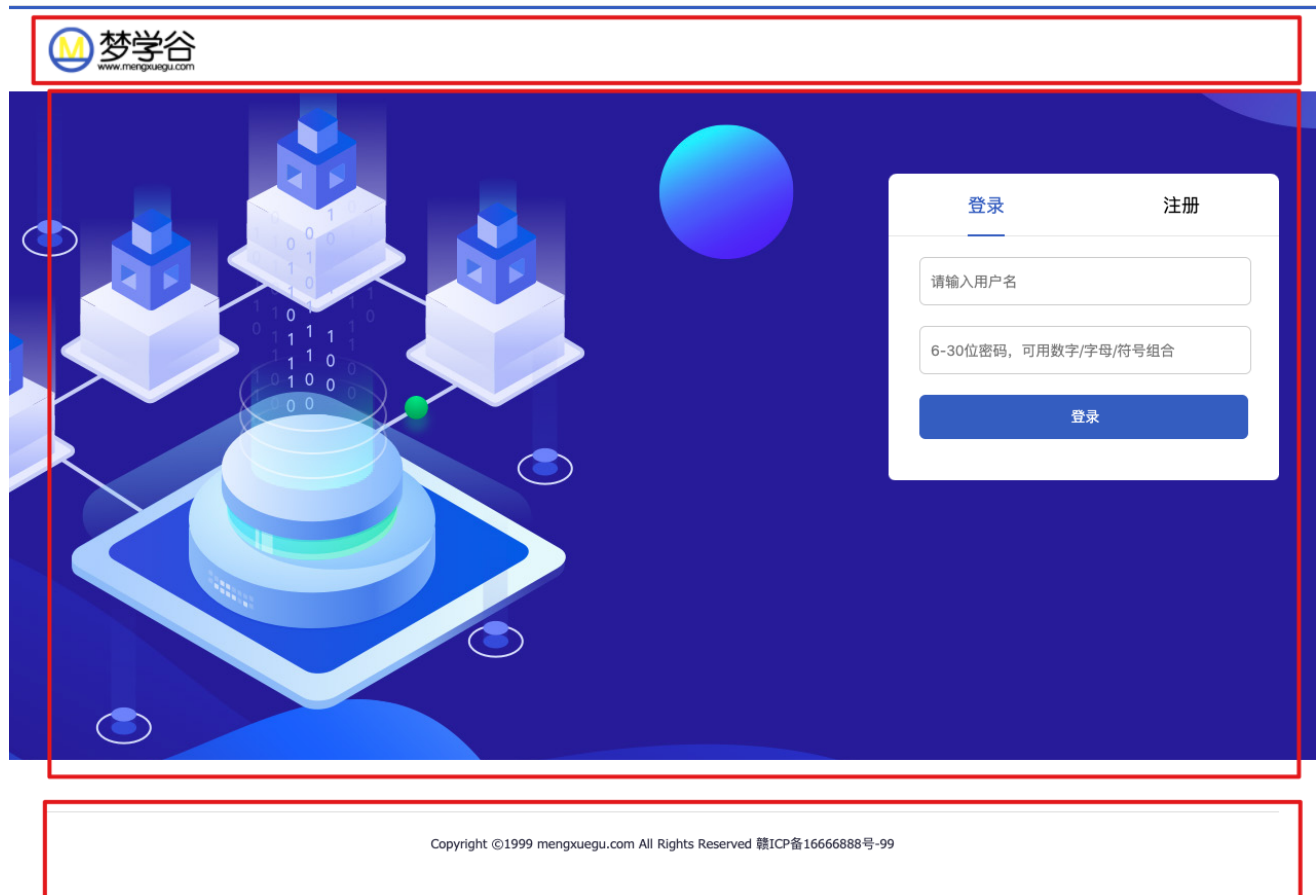
```
npm run dev
```

第三章 项目布局

<https://www.callmysoft.com/>

3.1 需求

布局采用3部分组成：头部、中间主区域、底部



3.2 头部区域组件

1. 创建头部区域组件：mengxuegu-auth-center/src/components/layout/AppFooter/index.vue

```
<template>
  <div class="mxg-header">
    <div class="logo">
      <a href="//www.mengxuegu.com" title="梦学谷">
        
      </a>
    </div>
  </div>
</template>
<style scoped>
.mxg-header {
  width: 100%;
  height: 80px;
  border-top: 3px solid #345dc2;
  z-index: 10;
}
```

闪电PDF编辑器

<https://www.callmysoft.com/>

```
.logo {
  width: 1200px;
  margin: 0 auto; /* 居中 */
  overflow: hidden;
  margin-top: 15px;
}
</style>
```

3.3 底部区域组件

1. 创建底部区域组件：mengxuegu-auth-center/src/components/layout/AppHeader/index.vue

```
<template>
  <!-- 底部 -->
  <div class="mxg-footer">
    <div class="footer-info">
      Copyright &copy;1999 mengxuegu.com &nbsp;All Rights Reserved&nbsp;
      <a href="http://www.beian.miit.gov.cn/" target="_blank" rel="nofollow"> 赣ICP备
      16666888号-99</a>
    </div>
  </div>
</template>

<script>
export default {
}
</script>
<style scoped>
/* 底部 */
.mxg-footer {
  width: 1200px;
  margin: 0 auto; /* 居中 */
  line-height: 60px;
  border-top: 1px solid #ddd;
}
.footer-info {
  text-align: center;
  font-size: 13px;
  color: #2C2C40;
}
.footer-info a {
  color: #2C2C40;
  text-decoration: none;
}
</style>
```

闪电PDF编辑器

<https://www.callmysoft.com/>

3.4 布局组件

1. 创建 `src\components\layout\index.vue` 文件定义布局结构，引入头部、底部组件，使用 `<router-view>` 渲染主区域组件

```
<template>
  <div >
    <app-header></app-header>
    <div class="mxg-main">
      <!-- 主区域组件渲染 -->
      <router-view></router-view>
    </div>
    <app-footer></app-footer>
  </div>
</template>

<script>
import AppHeader from '@components/layout/AppHeader'
import AppFooter from '@components/layout/AppFooter'
export default {
  components: { AppHeader, AppFooter },
}
</script>

<style >
.mxg-main {
  /* 自动计算高度 100vh 整屏高度-（头部高83+底部高61） */
  /* min-height: calc(100vh - 144px); */
  min-height: 700px;
  width: 100%;
}
</style>
```

3.5 App.vue 路由渲染出口

1. 在 App.vue 中定义所有路由渲染出口，注意样式保持一致


```
<template>
  <div id="app">
    <router-view></router-view>
  </div>
</template>

<style>
body {
  margin: 0;
  padding: 0px;
  font-family: "微软雅黑";
}
</style>
```

第四章 登录&注册组件与路由配置

4.1 登录&注册组件

1. 查看登录与注册页面组件 mengxuegu-auth-center/src/views/auth/login.vue

登录模板 login.vue 已经在项目中已经定义好了，直接使用。

4.2 路由配置

1. 配置路由，将登录页面渲染在主区域

- o 安装 Vue-Router 模块

```
npm i vue-router
```

- o 创建路由配置文件：mengxuegu-auth-center/src/router/index.js

```
import Vue from 'vue'
import Router from 'vue-router'

Vue.use(Router)

const router = new Router({
  mode: 'history',
  routes: [
    {
      path: '/',
      component: () => import('@/components/layout'),
      children: [
        {
          path: '',
          component: () => import('@/views/auth/login'),
        }
      ]
    }
  ]
})
```

闪电PDF编辑器

```
    }  
  ]  
},  
]  
}))  
  
export default router
```

- 在 mengxuegu-auth-center/src/main.js 将 router 路由对象添加到 Vue 实例中，顺便把 Vuex 状态管理 store/index.js 也添加中 Vue 实例中。

```
import Vue from 'vue'  
import App from './App.vue'  
import router from './router' // ++++  
import store from './store' // ++++  
  
Vue.config.productionTip = false  
  
new Vue({  
  router, // ++++  
  store, // ++++  
  render: h => h(App),  
}).$mount('#app')
```

- 浏览器请求 <http://localhost:7000>

第五章 封装 Axios 与 Mock.js

5.1 整合 Axios

安装 Axios，来向后台发送接口请求

- 安装 Axios 发送接口请求

```
npm install axios
```

- 自定义创建 axios 对象。创建 mengxuegu-auth-center/src/utlis/request.js

```
import axios from 'axios'  
  
const service = axios.create({  
  // .env.development 和 .env.productiont  
  baseURL: process.env.VUE_APP_BASE_API, // url = base url + request url
```

```
    timeout: 10000 // request timeout
  })

  // 请求拦截器
  service.interceptors.request.use(
    config => {
      return config
    },
    error => {
      return Promise.reject(error)
    }
  )

  // 响应拦截器
  service.interceptors.response.use(
    response => {
      // 正常响应
      const res = response.data
      return res
    },
    error => {
      // 响应异常
      return Promise.reject(error)
    }
  )

  export default service
```

5.2 对接 Mock.js 模拟数据接口

Mock.js 解决什么问题

问题：

前后端分离项目，前端和后端人员都是根据 API 文档进行开发项目的，不应该直接相互依赖，前端人员不应该等待后端开发好接口后再进行测试，既然不依赖后端接口，那前端人员应该如何测试呢？

解决：

可以通过模拟数据生成器，通过一定规则（API文档）生成模拟数据接口，提供给前端人员进行测试。

什么是 Mock.js

- 官网：<http://mockjs.com/>
- 文档：<https://github.com/nuysoft/Mock/wiki>
- Mock.js 是用于生成随机数据，拦截 Ajax 请求。

通过拦截 Ajax 请求，根据数据模板生成并返回模拟数据，让前端攻城师独立于后端进行开发，帮助编写单元测试。

闪电PDF编辑器

<https://www.callmysoft.com/>

什么是 EasyMock

Easy Mock 是一个可视化，并且能快速生成模拟数据的服务。是杭州大搜车无线团队出品的一个极其简单、高效、可视化、并且能快速生成模拟数据的在线 Mock 服务。

现在 Easy Mock 内置了 Mock.js，我们可以更愉快的伪造数据了。

官网: <https://www.easy-mock.com/> 服务器不稳定，访问不了

文档: <https://www.easy-mock.com/docs>

梦学谷搭建的EasyMock: <http://mock.mengxuegu.com/>

EasyMock 创建项目

1. 访问 <http://mock.mengxuegu.com/>，进行登录。

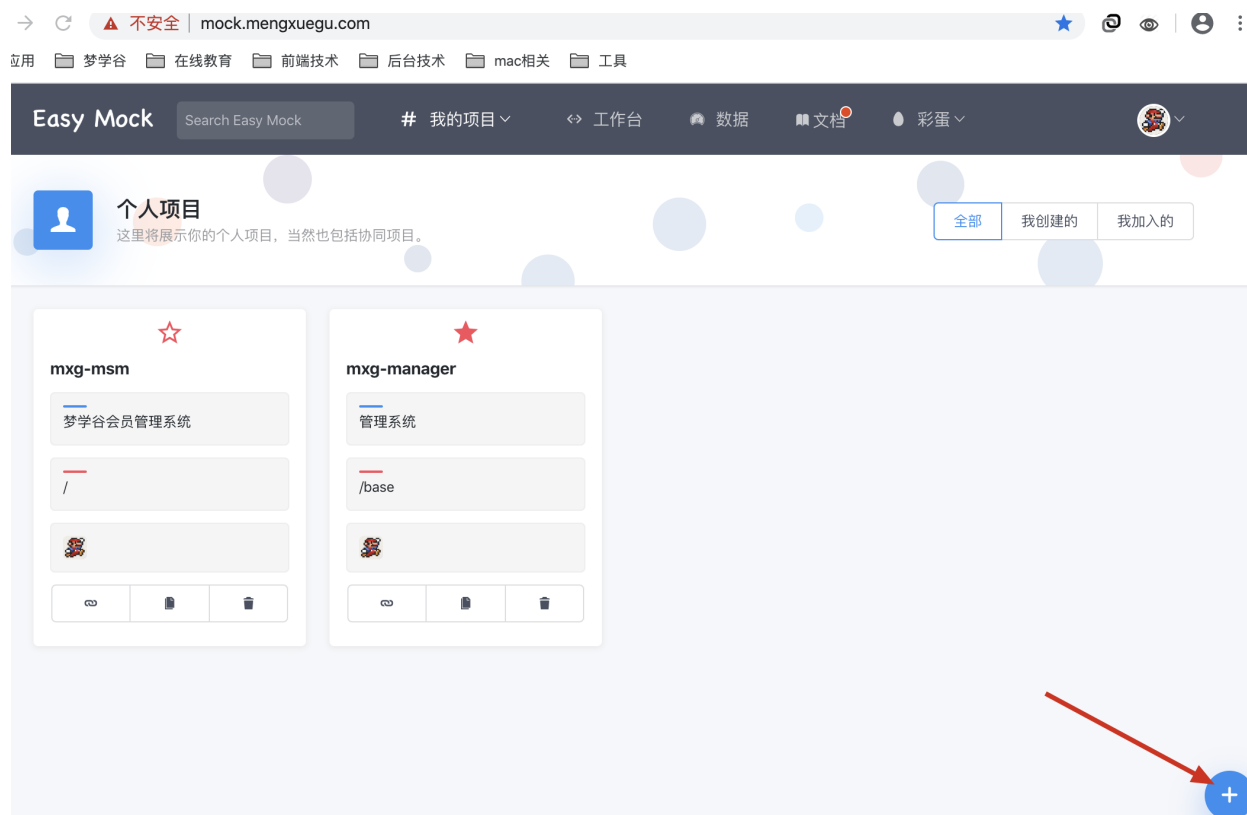
注意：没有单独的注册页面，第一次输入的用户名会自动帮你注册。特别注意没有修改密码的入口，所以一定要记住密码



2. 点击右下角 ，创建一个项目

闪电PDF编辑器

<https://www.callmysoft.com/>



3. 创建一个项目，如下：

闪电PDF编辑器

<https://www.callmysoft.com/>

归属 / 项目名 ?

1586841388312 ▼

/ 梦学谷前后端分离单点登录

项目基础 URL ?

/ mxg-ss0

项目描述

梦学谷前后端分离单点登录

Swagger Docs API (可选)

URL ▼

http://example.com/swagger.json

如果后台有提供 Swagger 文档（并且没有验证授权的问题），于是我们可以在此处填写 Swagger 的接口地址，Easy Mock 会自动基于此接口创建 Mock 接口。 ?

邀请成员 协同编辑 (可选)

用户昵称、用户名，支持模糊匹配

创建

EasyMock 添加登录接口

1. 进入项目，点击 **创建接口**

闪电PDF编辑器

<https://www.callmysoft.com/>



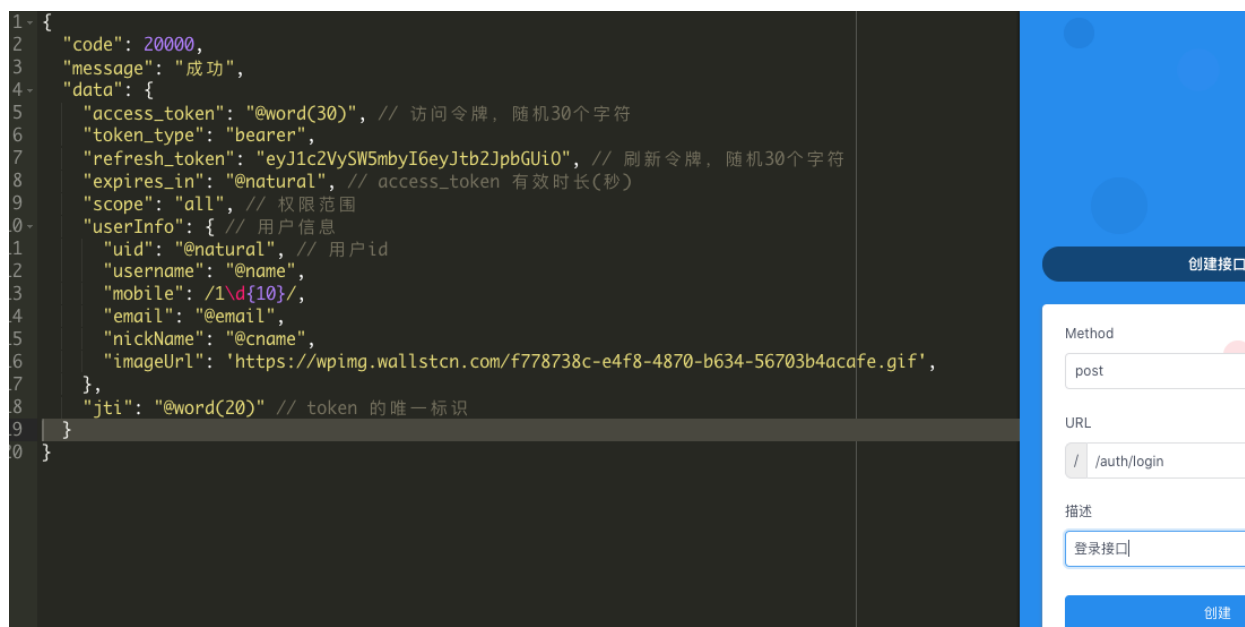
2. 添加登录接口

- 请求URL: `/auth/login`
- 请求方式: `post`
- 描述: 登录接口
- mock.js 配置:

```
{
  "code": 20000,
  "message": "成功",
  "data": {
    "access_token": "@word(30)", // 访问令牌, 随机30个字符
    "token_type": "bearer",
    "refresh_token": "eyJ1c2VySW5mbyI6eyJtb2JpbGUiO", // 刷新令牌, 随机30个字符
    "expires_in": "@natural", // access_token 有效时长(秒)
    "scope": "all", // 权限范围
    "userInfo": { // 用户信息
      "uid": "@natural", // 用户id
      "username": "@name",
      "mobile": "/1\d{10}/",
      "email": "@email",
      "nickName": "@cname",
      "imageUrl": 'https://wpimg.wallstcn.com/f778738c-e4f8-4870-b634-56703b4acafe.gif',
    },
    "jti": "@word(20)" // token 的唯一标识
  }
}
```

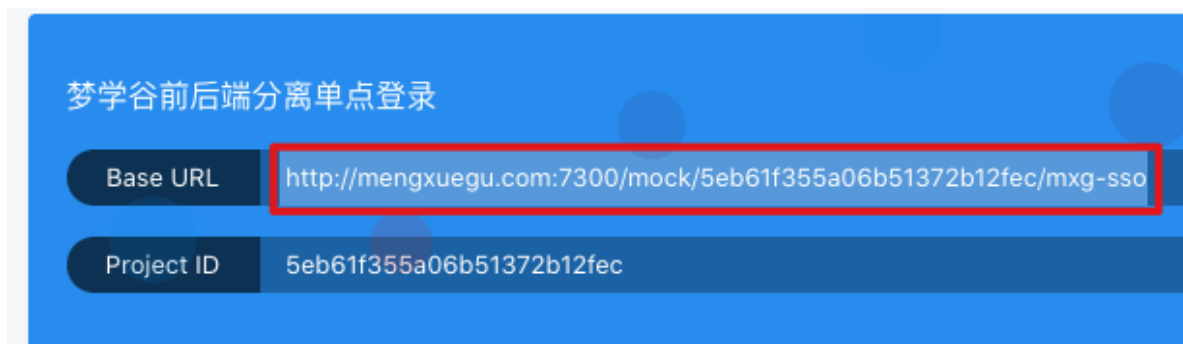
闪电PDF编辑器

<https://www.callmysoft.com/>



3. 修改接口地址,

- 复制后台接口 BASE URL , 如下:



- 找到项目根目录下的 `.env.development` 的 `VUE_APP_SERVICE_URL` 修改为如下:

```
# 定义请求的基础URL, 方便跨域请求时使用
VUE_APP_BASE_API = '/dev-api'

# 接口服务地址, 以你自己的为主
VUE_APP_SERVICE_URL = 'http://mengxuegu.com:7300/mock/5eb61f355a06b51372b12fec/mxg-sso'
```

第六章 登录功能实现

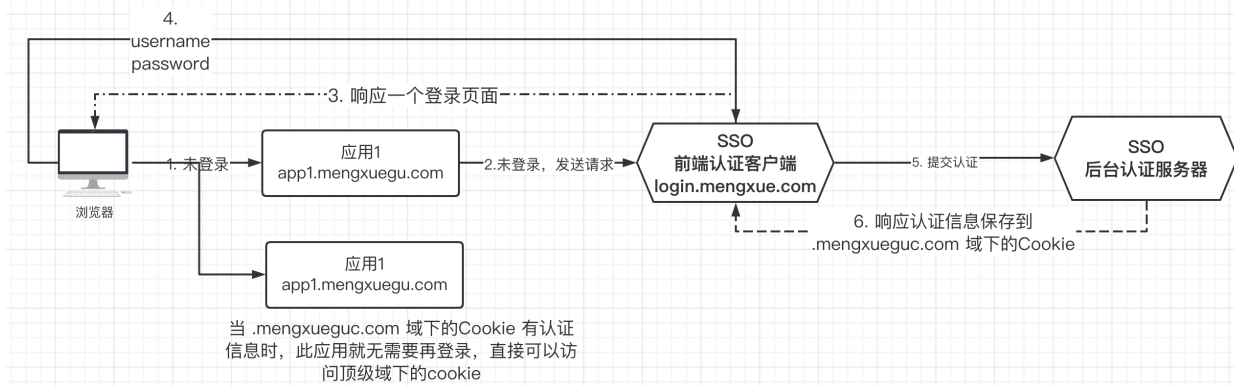
6.1 登录认证流程

单点登录认证流程图

闪电PDF编辑器

<https://www.callmysoft.com/>

梦学谷 单点登录流程



1. 门户客户端要求登录时, 输入用户名密码, 认证客户端提交数据给认证服务器
2. 认证服务器校验用户名密码是否合法, 合法响应用户基本令牌 userInfo、访问令牌 access_token、刷新令牌 refresh_token。不合法响应错误信息。

定义 Api 调用登录接口

1. 创建调用认证API接口文件 src/api/auth.js

登录时, 要在请求头带上客户端ID和客户端密码, 并且在请求头指定数据格式

```

import request from '@/utils/request'

// 数据格式
const headers = { 'Content-Type': 'application/x-www-form-urlencoded' }

// 请求头添加 Authorization: Basic client_id:client_secret
const auth = {
  username: 'mxg-blog-admin', // client_id
  password: '123456' // client_secret
}

// 登录, 获取 token 接口
export function login(data) {
  return request({
    headers,
    auth,
    url: '/auth/login',
    method: 'post',
    params: data
  })
}

```

Vuex 登录信息状态管理

闪电PDF编辑器

<https://www.callmysoft.com/>

当登录成功后，后台响应的 userInfo、access_token、refresh_token 信息使用 Vuex 进行管理，并且将这些信息保存到浏览器 Cookie 中。

1. 安装 js-cookie 和 vuex 模块

```
npm install --save js-cookie vuex
```

2. 在 mengxuegu-auth-center/src/store/index.js 创建 Vuex.Store 实例，导入 ./modules/auth.js 状态模块

```
import Vue from 'vue'
import Vuex from 'vuex'
import auth from './modules/auth' // auth 状态模块

Vue.use(Vuex)

const store = new Vuex.Store({
  modules: {
    auth
  }
})

export default store
```

3. 检查 mengxuegu-auth-center/src/main.js 是否将 store 已添加到Vue 实例中

闪电PDF编辑器

<https://www.callmysoft.com/>

```

import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'

Vue.config.productionTip = false

new Vue({
  router,
  store,
  render: h => h(App),
}).$mount('#app')

```

4. 创建认证状态模块文件 `src/store/modules/auth.js` 中添加对 `userInfo`、`access_token`、`refresh_token` 状态的管理

```

import { login } from '@api/auth'
import { PcCookie, Key } from '@utils/cookie' // 对 cookie 操作

// 定义状态, state必须是function
const state = {
  userInfo: PcCookie.get(Key.userInfoKey)
    ? JSON.parse(PcCookie.get(Key.userInfoKey)) : null, // 用户信息对象
  accessToken: PcCookie.get(Key.accessTokenKey), // 访问令牌字符串
  refreshToken: PcCookie.get(Key.refreshTokenKey), // 刷新令牌字符串
}

// 改变状态值
const mutations = {

  // 赋值用户状态
  SET_USER_STATE (state, data) {
    console.log('SET_USER_STATE', data)
    // 状态赋值
    const { userInfo, access_token, refresh_token } = data
    state.userInfo = userInfo
    state.accessToken = access_token
  }
}

```

```

state.refreshToken = refresh_token

// 保存到cookie中
PcCookie.set(Key.userInfoKey, userInfo)
PcCookie.set(Key.accessTokenKey, access_token)
PcCookie.set(Key.refreshTokenKey, refresh_token)
},

// 重置用户状态,退出和登录失败时用
RESET_USER_STATE (state) {
  // 状态置空
  state.userInfo = null
  state.accessToken = null
  state.refreshToken = null

  // 移除cookie
  PcCookie.remove(Key.userInfoKey)
  PcCookie.remove(Key.accessTokenKey)
  PcCookie.remove(Key.refreshTokenKey)
}
}

// 定义行为
const actions = {

  // 登录操作      ++++++ 4.
  UserLogin ({ commit }, userInfo) {
    const { username, password } = userInfo
    return new Promise((resolve, reject) => {
      // 调用登录接口 /api/auth.js#login
      login({ username: username.trim(), password: password }).then(response => {
        // 获取响应值
        const { code, data } = response
        if(code === 20000) {
          // 状态赋值
          commit('SET_USER_STATE', data)
        }
        resolve(response) // 不要少了
      }).catch(error => {
        // 重置状态
        commit('RESET_USER_STATE')
        reject(error)
      })
    })
  }

}

export default {
  state,
  mutations,
  actions
}

```

闪电PDF编辑器

<https://www.callmysoft.com/>

5. 查看 utils/cookie.js 设置了保存的时长与域，对应域设置在 .env.development 和 .env.production 文件里的

```
# cookie保存的域名, utils/cookie.js 要用
VUE_APP_COOKIE_DOMAIN = 'location'
```

提交登录触发 action

1. 在登录页 src/views/auth/login.vue 的 created 生命钩子里获取redirectURL，是引发跳转到登录页的引发跳转 URL，登录成功后需要重定向回 redirectURL

```
created() {
  // 判断URL上是否带有redirectURL参数
  if(this.$route.query.redirectURL) {
    this.redirectURL = this.$route.query.redirectURL
  }
},

methods: {
}
```

2. 修改 src/views/auth/login.vue 的 loginSubmit 方法，触发 store/modules/auth.js 中的 UserLogin 进行登录。并导入 @/utils/validate 正则表达式校验用户名是否合法。

```
import {isvalidUsername} from '@/utils/validate' // 校验规则

export default {

  methods: {
    // 提交登录
    loginSubmit() {
      // 如果在登录中, 不允许登录
      if(this.subState) {
        return false;
      }
      if(!isvalidUsername(this.loginData.username)) {
        this.loginMessage = '请输入正确用户名'
        return false
      }

      if (this.loginData.password.length < 6) {
        this.loginMessage = '请输入正确的用户名或密码';
        return false;
      }

      this.subState = true // 提交中
```

闪电PDF编辑器

```

// 提交登录，不要以 / 开头
this.$store.dispatch('UserLogin', this.loginData).then((response) => {
  const { code, message } = response
  if(code === 20000) {
    // 跳转回来源页面
    window.location.href = this.redirectURL
  }else {
    this.loginMessage = message
  }
  this.subState = false // 提交完
}).catch(error => {
  // 进度条结束
  this.subState = false // 提交完
  this.loginMessage = '系统繁忙，请稍后重试'
})
},
},
},

```

6.2 测试

登录成功后，重定向回到redirectURL参数值对应的页面，如果不带 redirectURL 重写向到 mengxuegu.com

第七章 注册功能实现

7.1 EasyMock 添加注册接口

1. 查询用户名是否已被注册

- 请求URL: `/system/api/user/username/{username}`
- 请求方式: `get`
- 描述: 查询用户名是否已被注册
- mock.js 配置:

```

{
  "code": 20000,
  "message": "查询成功",
  "data": true // true 已被注册, false 未被注册
}

```

2. 添加注册接口

- 请求URL: `/system/api/user/register`
- 请求方式: `post`
- 描述: 注册接口
- mock.js 配置:

闪电PDF编辑器

<https://www.callmysoft.com/>

```
{
  "code": 20000,
  "message": "注册成功"
}
```

7.2 定义 Api 调用用户协议和注册接口

在 src/api/auth.js 添加 获取用户协议 和 提交注册 接口方法，

用户协议内容已经在 `mengxuegu-auth-center/public/xieyi.html` 存在了，直接调用它即可。

```
// 获取协议内容
export function getXieyi() {
  return request({
    url: `${window.location.href}/xieyi.html`, // 访问到的是 public/xieyi.html
    method: 'get'
  })
}

// 查询用户名是否已被注册
export function getUserByUsername(username) {
  return request({
    url: `/system/api/user/username/${username}`,
    method: 'get'
  })
}

// 提交注册数据
export function register(data) {
  return request({
    headers,
    auth,
    url: `/system/api/user/register`,
    method: 'post',
    params: data
  })
}
```

7.3 提交注册数据

在 mengxuegu-auth-center/src/views/auth/login.vue 添加如下内容：

1. 导入 `getXieyi` 方法，并 `created` 钩子中调用api接口获取协议内容

```
<script>
import {isvalidUsername} from '@utils/validate'
```

闪电PDF编辑器

```
// 导入 `getXieyi` 方法
import {getXieyi, getUserByUsername, register} from '@api/auth'

export default {
  // 加上 async
  async created() {
    // 判断URL上是否带有redirectURL参数
    if(this.$route.query.redirectURL) {
      this.redirectURL = this.$route.query.redirectURL
    }
    // 获取协议内容
    this.xieyiContent = await getXieyi()
  },
}
</script>
```

2. 在 regSubmit 方法中提交注册表单数据

注意：regSubmit 方法前面加上 **async** 关键字

```
// 提交注册
async regSubmit() { // 前面加上 async 关键字
  // 如果在登录中，不允许登录
  if(this.subState) {
    return false;
  }

  if( !isValidUsername(this.registerData.username) ) {
    this.regMessage = '请输入4-30位用户名，中文、数字、字母和下划线'
    return false
  }

  // 校验用户名是否存在
  const { code, message, data } =
    await getUserByUsername(this.registerData.username)
  // 不为 20000,则后台校验用户名有问题
  if( code !== 20000 ) {
    this.regMessage = message
    return false
  }
  if( data ) { // data是 true 已被注册, false未被注册
    this.regMessage = '用户名已被注册, 请重新输入用户名'
    return false
  }

  if (this.registerData.password.length < 6 ||
    this.registerData.password.length > 30) {
    this.regMessage = '请输入6-30位密码,区分大小写且不可有空格'
    return false
  }
}
```

闪电PDF编辑器

<https://www.callmysoft.com/>


```

    if (this.registerData.password !== this.registerData.repPassword) {
      this.regMessage = '两次输入密码不一致'
      return false
    }

    if (!this.registerData.check) {
      this.regMessage = '请阅读并同意用户协议'
      return false
    }

    this.subState = true // 提交中

    // 提交注册
    register(this.registerData).then(response => {
      this.subState = false
      const {code, message} = response
      if(code === 20000) {
        // 注册成功, 切换登录页
        this.changetab(1)
      }else {
        this.regMessage = message
      }
    }).catch(error => {
      this.subState = false
      this.regMessage = '系统繁忙, 请稍后重试'
    })
  }
}

```

7.4 测试

1. 提交注册, 会提示 **用户名已被注册, 请重新输入用户名**,
2. 修改 `/system/api/user/username/{username}` 接口的 data 的值为 false 表示未注册

```

{
  "code": 20000,
  "message": "查询成功",
  "data": false // true 已被注册, false 未被注册
}

```

3. 注册成功后, 切换到登录标签页

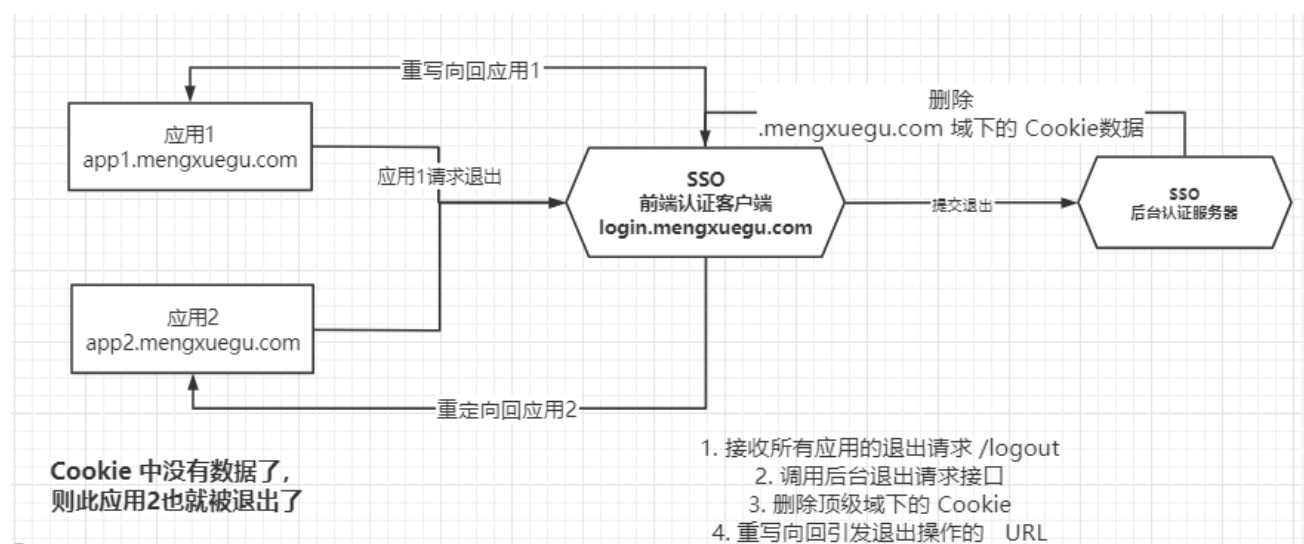
第八章 单点退出系统

8.1 需求分析

闪电PDF编辑器

<https://www.callmysoft.com/>

所有应用系统退出，全部发送请求到当前认证中心进行处理，发送请求后台删除用户登录数据，并将 cookie 中的用户数据清除。



8.2 EasyMock 添加退出系统模拟接口

- 请求URL: `/auth/logout`
- 请求方式: `get`
- 描述: 退出系统
- mock.js 配置:

```
{  
  "code": 20000,  
  "message": "退出成功"  
}
```

8.3 定义Api调用退出接口

1. 在 api/auth.js 定义调用退出接口

```
// 退出系统  
export function logout(accessToken) {  
  return request({  
    url: '/auth/logout',  
    method: 'get',  
    params: {  
      accessToken  
    }  
  })  
}
```

闪电PDF编辑器

<https://www.callmysoft.com/>

8.4 定义 Vuex 退出行为

1. 在 `src/store/modules/login.js` 状态管理文件中的 actions 对象中添加调用 logout 退出api方法。退出成功后回到登录页。

```
// 1. 导入 logout , ++++++
import { login, logout } from '@/api/login'

// 定义行为
const actions = {

  // 2. 退出, ++++++
  UserLogout({ state, commit }, redirectURL) {
    // 调用退出接口, 上面不要忘记导入 logout 方法
    logout(state.accessToken).then(() => {
      // 重置状态
      commit('RESET_USER_STATE')
      // // 退出后, 重写向地址, 如果没有传重写向到登录页 /
      window.location.href = redirectURL || '/'
    }).catch(() => {
      // 重置状态
      commit('RESET_USER_STATE')
      window.location.href = redirectURL || '/'
    })
  }
}
```

8.5 路由拦截器退出操作

1. 应用系统访问 <http://localhost:7000/logout?redirectURL=xxx> 进行退出, 我们添加路由前置拦截 /logout 路由请求进行调用 UserLogout 进行退出操作。

```
import Vue from 'vue'
import Router from 'vue-router'

Vue.use(Router)

const router = new Router({
  mode: 'history',
  routes: [
    {
      path: '/',
      component: () => import('@/components/layout'),
      children: [
        {
          path: ''
```

闪电PDF编辑器

```

        component: () => import('@/views/auth/login'),
      },
    ],
  },
]
})

// 导入vuex状态对象store ++++++
import store from '@/store'

// 路由拦截器 ++++++
router.beforeEach((to, from, next) => {
  if(to.path === '/logout') {
    // 退出
    store.dispatch('UserLogout', to.query.redirectURL)
  } else {
    next()
  }
})

export default router

```

2. 测试

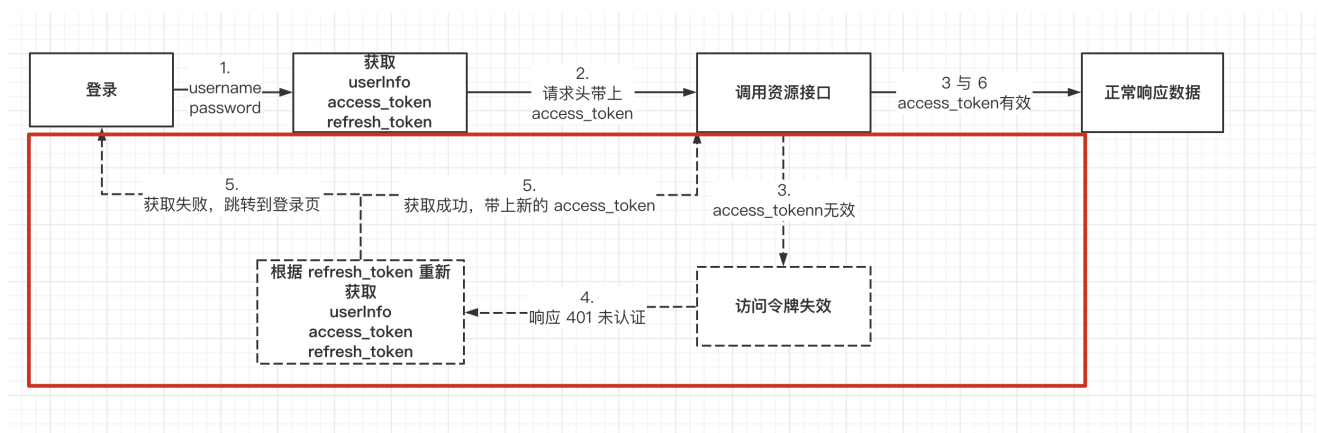
访问: <http://localhost:7000/logout?redirectURL=http://www.mengxuegu.com>

查看: 浏览器 cookie 没有值

第九章 刷新令牌获取新令牌

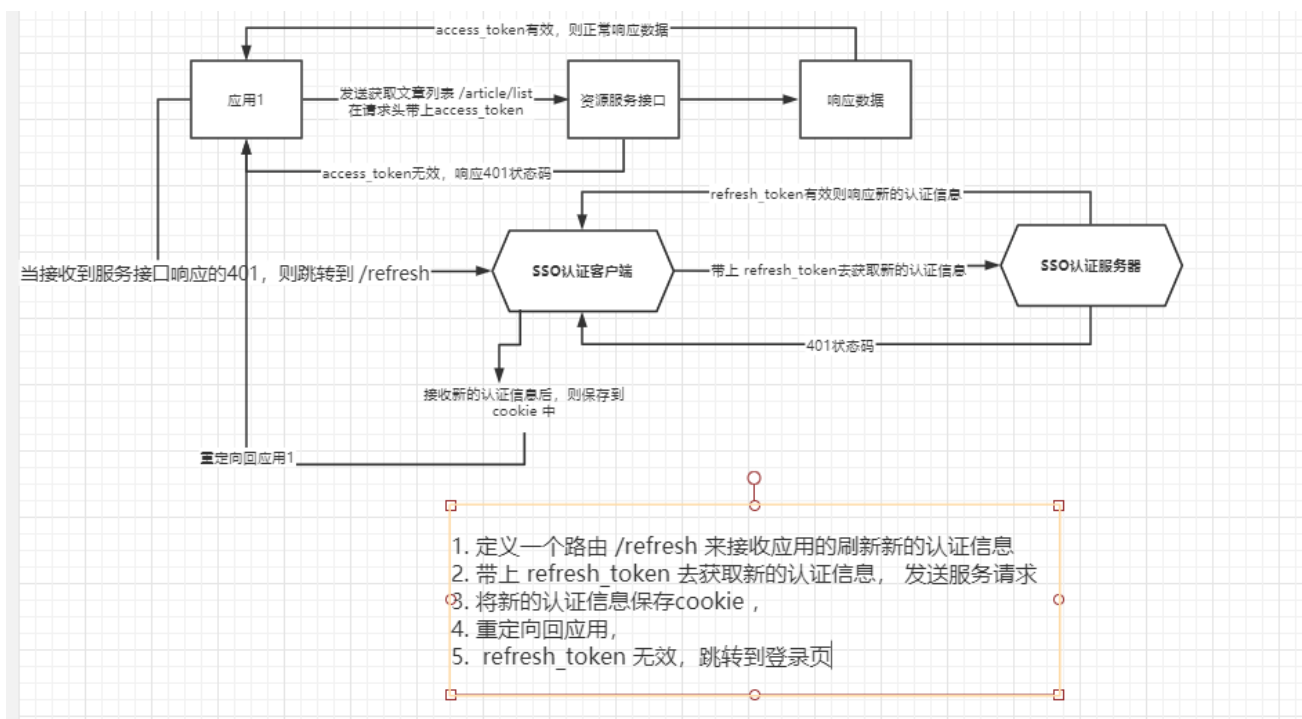
当应用系统请求后台资源接口时, 要在请求头带上 accessToken 去请求接口, 如果 accessToken 有效, 资源服务器正常响应数据。

如果访问令牌 accessToken 过期, 资源服务器会响应 401 状态码。当应用系统接收到 401 状态码时, 通过刷新令牌 refreshToken 获取去请求新令牌完成新的重新身份。



闪电PDF编辑器

<https://www.callmysoft.com/>



9.1 创建刷新令牌组件

在认证前端 mengxuegu-auth-center 创建一个刷新组件, 用于接收应用系统发送请求到认证前端, 进行刷新令牌重新身份认证。

刷新组件以弹窗方式: 提示正在重新身份认证

1. 创建组件模板 mengxuegu-auth-center/src/views/auth/refresh.vue

```

<template>
  <div>
    <!-- 弹窗 -->
    <div v-show="visible" >
      <!--这里是要展示的内容层-->
      <div class="content">
        <span v-html="message"></span>
      </div>
      <!--半透明背景层-->
      <div class="over"></div>
    </div>
  </div>
</template>

```

2. 添加模板样式

参考: 03-配套资料/前端单点登录资料/css/refresh.css

<style copied>

闪电PDF编辑器

<https://www.callmysoft.com/>

```

.content {
  position: fixed;
  height: 120px;
  width: 500px;
  line-height: 120px;
  text-align: center;
  font-size: 19px;
  color: #303133;
  background-color: #fff;
  border-radius: 0.25rem;
  left: 50%;
  top: 30%;
  transform: translate(-50%, -50%);
  z-index: 1000;
}
a {
  color: #345dc2;
  text-decoration: none;
}
a:hover {
  text-decoration: underline;
}
.over {
  position: fixed;
  width: 100%;
  height: 100%;
  opacity: 0.5; /* 透明度为50% */
  filter: alpha(opacity=50);
  top: 0;
  left: 0;
  z-index: 999;
  background-color: #000;
}
</style>

```

3. data选项中声明变量， created 钩子中获取重写向URL， 和发送请求刷新身份

```

<script >
  export default {
    data () {
      return {
        visiabe: 1, // 1 打开弹窗, 0 关闭弹窗
        message: '请稍等, 正在重新身份认证...',
        redirectURL: null
      }
    },

    created () {
      this.redirectURL = this.$route.query.redirectURL || '/'
      this.refreshLogin()
    },

```

闪电PDF编辑器

<https://www.callmysoft.com/>

```
methods: {  
  // 刷新令牌登录  
  refreshLogin () {  
  
  }  
}  
};  
</script>
```

9.2 添加刷新组件路由配置

1. 在 mengxuegu-auth-center/src/router/index.js 添加刷新组件的路由配置

```
const router = new Router({  
  mode: 'history',  
  routes: [  
    {  
      path: '/',  
      component: () => import('@components/layout'),  
      children: [  
        {  
          path: '',  
          component: () => import('@views/auth/login'),  
        }  
      ],  
    },  
    // 刷新组件路由配置 +++++  
    {  
      path: '/refresh',  
      component: () => import('@components/layout'),  
      children: [  
        {  
          path: '',  
          component: () => import('@views/auth/refresh'),  
        }  
      ],  
    }  
  ],  
})
```

2. 测试:

访问 <http://localhost:7000/refresh?redirectURL=http://localhost:3000/>

9.3 EasyMock 添加刷新令牌接口

- 请求URL: `/auth/user/refreshToken`

- 请求方式: `get`
- 描述: 刷新令牌接口
- mock.js 配置: 刷新令牌获取的身份信息和 登录时获取的是一样的。

```
{
  "code": 20000,
  "message": "成功",
  "data": {
    "access_token": "@word(30)", // 访问令牌, 随机30个字符
    "token_type": "bearer",
    "refresh_token": "eyJ1c2VySW5mbyI6eyJtb2JpbGUiO", // 刷新令牌, 随机30个字符
    "expires_in": "@natural", // access_token 有效时长(秒)
    "scope": "all", // 权限范围
    "userInfo": { // 用户信息
      "uid": "@natural", // 用户id
      "username": "@name",
      "mobile": "/1\d{10}/",
      "email": "@email",
      "nickName": "@cname",
      "imageUrl": 'https://wpimg.wallstcn.com/f778738c-e4f8-4870-b634-56703b4acafe.gif',
    },
    "jti": "@word(20)" // token 的唯一标识
  }
}
```

9.4 定义 Api 调用刷新令牌接口

1. 添加调用 **刷新令牌获取新令牌接口** API 方法, 在 `mengxuegu-auth-center/src/api/auth.js`

```
// 刷新令牌接口 ++++++
export function refreshToken (refreshToken) {
  return request({
    headers,
    auth,
    url: `/auth/user/refreshToken`,
    method: 'get',
    params: {
      refreshToken
    }
  })
}
```

9.5 Vuex 发送请求与重置状态

1. `store/modules/login.js` 添加如下代码:

闪电PDF编辑器

<https://www.callmysoft.com/>

- o 导入 refreshToken
- o actions 中 添加发送刷新令牌请求 行为

```
// 1. 导入 refreshToken +++++
import { login, logout, refreshToken } from '@api/auth'
import { PcCookie, Key } from '@utils/cookie' // 对 cookie 操作

// 省略。。。

// 定义行为
const actions = {

  // 2. 发送刷新令牌 +++++
  SendRefreshToken({ state, commit }) {
    return new Promise((resolve, reject) => {
      // 判断是否有刷新令牌
      if(!state.refreshToken) {
        commit('RESET_USER_STATE')
        reject('没有刷新令牌')
        return
      }
      // 发送刷新请求
      refreshToken(state.refreshToken).then(response => {
        // console.log('刷新令牌新数据', response)
        // 更新用户状态新数据
        commit('SET_USER_STATE', response.data)
        resolve() // 正常响应钩子
      }).catch(error => {
        // 重置状态
        commit('RESET_USER_STATE')
        reject(error)
      })
    })
  },
}
```

9.6 重构刷新令牌组件

1. 在 mengxuegu-auth-center/src/views/auth/refresh.vue 中的 refreshLogin 方法中触发 store/modules/auth.js 中的 SendRefreshToken 行为来完成刷新身份。

```
methods: {
  // 刷新令牌登录
  refreshLogin () {
    this.$store.dispatch('SendRefreshToken').then(response => {
      // this.message = '身份已认证, 正在为您进行页面跳转.....'
      // 刷新成功, 重写向回去
      window.location.href = this.redirectURL
    })
  }
}
```

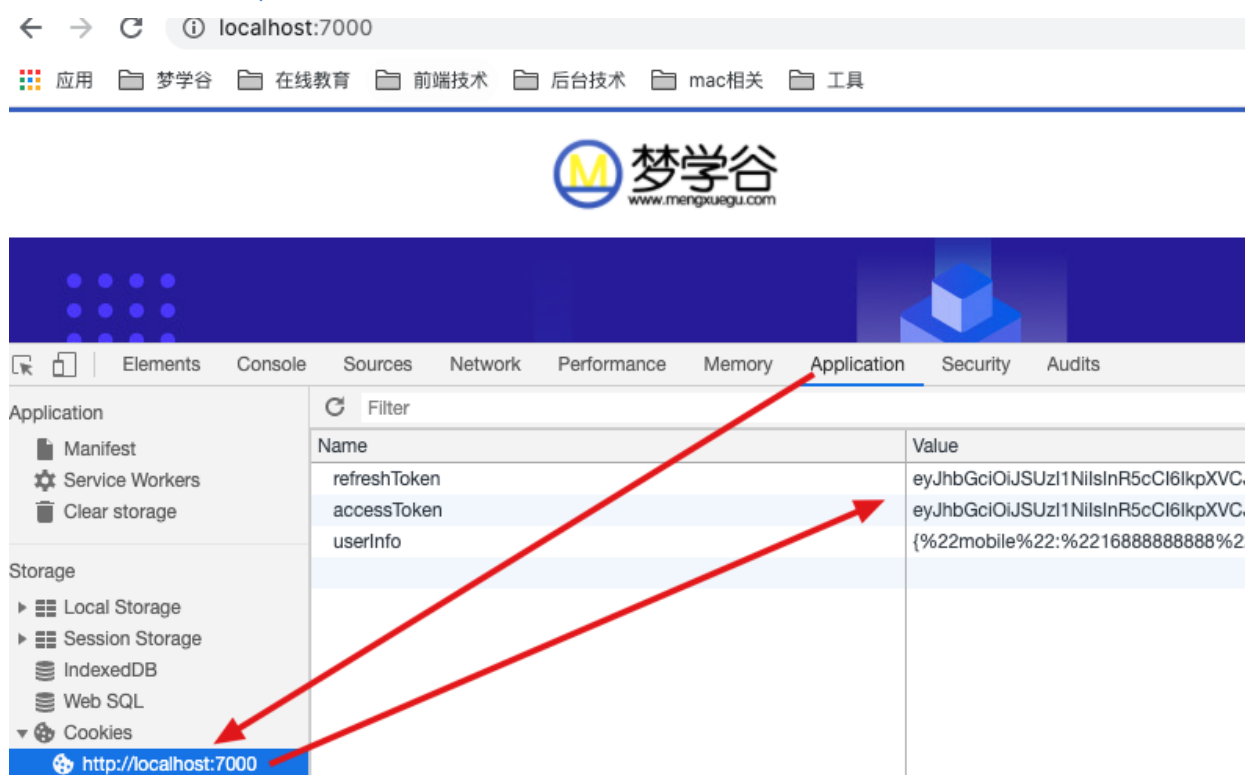
```

    }).catch(error => {
      // 刷新失败，去登录页
      this.message =
        `您的身份已过期，请点击

```

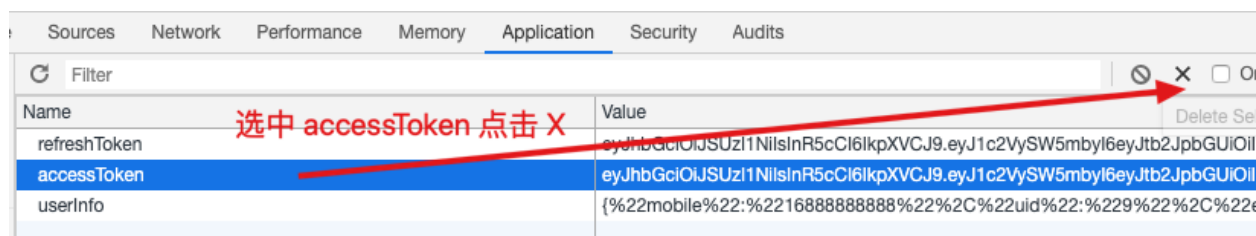
9.7 测试

1. 重启 mengxuegu-auth-center 项目
2. 访问认证登录页 <http://localhost:7000/>，进行正常登录。
3. 登录后，再次访问 <http://localhost:7000/> 登录页，打开浏览器控制台确保 Cookie 中有值



4. 将 Cookie 中的 accessToken 删掉，认为 accessToken 已经过期了，就可以刷新令牌了。

注意是点击 X，如果点错了重新登录下。



闪电PDF编辑器

<https://www.callmysoft.com/>

5. 访问 <http://localhost:7000/refresh?redirectURL=http://localhost:3000/> 后，重定向回 <http://localhost:3000/> 并且cookie中又有访问令牌了。

| Sources Network Performance Memory Application Security Audits | |
|--|---|
| Filter | |
| name | value |
| refreshToken | eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySW5r |
| accessToken | eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySW5r |
| userInfo | {%22uid%22:%229%22%2C%22nickName%22:%22%E |

6. 如果你想看是否正常响应，可以把跳转 `window.location.href` 注释掉，向 `this.message` 添加提示信息。

```
methods: {
  // 刷新令牌登录
  refreshLogin () {
    this.$store.dispatch('SendRefreshToken').then(response => {
      this.message = '身份已认证，正在为您进行页面跳转.....'
      // 刷新成功，重写向回去
      // window.location.href = this.redirectURL
    }).catch(error => {
      // 刷新失败，去登录页
      this.message = `您的身份已过期，请点击
```

闪电PDF编辑器

<https://www.callmysoft.com/>