

Spring Authorization Server入门 (十八) Vue项目使用PKCE模式对接认证服务

叹雪飞花 2023-09-17 👁 685 ⌚ 阅读3分钟

关注

Vue单页面项目使用授权码模式对接流程说明

以下流程摘抄自官网

在本例中为授权代码流程。授权码流程的步骤如下：

1. 客户端通过重定向到授权端点来发起 OAuth2 请求。对于公共客户端，此步骤包括生成 `code_verifier` 并计算 `code_challenge`，然后将其作为查询参数发送。
2. 如果用户未通过身份验证，授权服务器将重定向到登录页面。身份验证后，用户将再次重定向回授权端点。
3. 如果用户未同意所请求的范围并且需要同意，则会显示同意页面。
4. 一旦用户同意，授权服务器会生成一个 `authorization_code` 并通过 `redirect_uri` 重定向回客户端。
5. 客户端通过查询参数获取 `authorization_code` 并向 [Token Endpoint](#) 发起请求。对于公共客户端，此步骤包括发送 `code_verifier` 参数而不是用于身份验证的凭据。

Vue项目中修改内容

安装crypto-js依赖

已安装可以忽略，该依赖是为了计算 `Code Challenge` 使用

TypeScript下额外添加 @types/crypto-js 依赖

▼

shell 复制代码

```
1 npm install @types/crypto-js
```

编写公共方法

编写 Code Verifier 生成与 Code Challenge 的计算方法

创建PKCE工具js文件

▼

js 复制代码

```
1 import CryptoJS from 'crypto-js'
2
3 /**
4  * 生成 CodeVerifier
5  *
6  * return CodeVerifier
7  */
8 export function generateCodeVerifier() {
9     return generateRandomString(32)
10 }
11
12 /**
13  * 生成随机字符串
14  * @param length 随机字符串的长度
15  * @returns 随机字符串
16  */
17 export function generateRandomString(length: number) {
18     let text = ''
19     const possible = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789'
20     for (let i = 0; i < length; i++) {
21         text += possible.charAt(Math.floor(Math.random() * possible.length))
22     }
23     return text
24 }
25
```

```
29 * @returns Code Challenge
30 */
31 export function generateCodeChallenge(code_verifier: string) {
32     return base64URL(CryptoJS.SHA256(code_verifier))
33 }
34
35 /**
36 * 将字符串base64加密后在转为url string
37 * @param str 字符串
38 * @returns base64转码后转为url string
39 */
40 export function base64URL(str: CryptoJS.lib.WordArray) {
41     return str
42         .toString(CryptoJS.enc.Base64)
43         .replace(/=/g, '')
44         .replace(/\+/g, '-')
45         .replace(/\//g, '_')
46 }
47
48 /**
49 * 将字符串加密为Base64格式的
50 * @param str 将要转为base64的字符串
51 * @returns 返回base64格式的字符串
52 */
53 export function base64Str(str: string) {
54     return CryptoJS.enc.Base64.stringify(CryptoJS.enc.Utf8.parse(str));
55 }
```

编写获取地址栏参数方法

略，已在上篇文章中贴出

编写请求Token方法

略，已在上篇文章中贴出

在环境变量配置文件中添加配置

js 复制代码

```
3 # PKCE流程使用的客户端Id
4 VITE_PKCE_CLIENT_ID=pkce-message-client
5 # 授权码模式使用的回调地址
6 VITE_PKCE_REDIRECT_URI=http://127.0.0.1:5173/PkceRedirect
```

创建处理回调的页面 PkceRedirect.vue

页面加载时会尝试从地址栏获取参数 `code`，如果能获取到说明是从认证服务回调过来的，执行换取token流程，如果没有获取到code说明需要发起授权申请。跟之前的授权码流程是一致的。

[js 复制代码](#)

```
1 <script setup lang="ts">
2 import router from '../router'
3 import { getToken } from '@api/Login'
4 import { getQueryString } from '@util/GlobalUtils'
5 import { createDiscreteApi } from 'naive-ui'
6 import { generateCodeVerifier, generateCodeChallenge } from '@util/pkce'
7
8 const { message } = createDiscreteApi(['message'])
9
10 // 生成CodeVerifier
11 let codeVerifier: string = generateCodeVerifier()
12 // codeChallenge
13 let codeChallenge: string = generateCodeChallenge(codeVerifier)
14 // 生成state
15 let state: string = generateCodeVerifier()
16
17 // 获取地址栏授权码
18 const code = getQueryString('code')
19
20 if (code) {
21   // 从缓存中获取 codeVerifier
22   const state = localStorage.getItem('state')
23   // 校验state, 防止cors
24   const urlState = getQueryString('state')
25   if (urlState !== state) {
26     message.warning('state校验失败.')
27   } else {
28     // 从缓存中获取 codeVerifier
29     const code_verifier = localStorage.getItem('codeVerifier')
30     getToken({
```

```
34     code,
35     code_verifier,
36     state
37   })
38   .then((res: any) => {
39     localStorage.setItem('accessToken', JSON.stringify(res))
40     router.push({ path: '/' })
41   })
42   .catch((e) => {
43     message.warning(`请求token失败: ${e.data.error || e.message || e.statusText}`)
44   })
45 }
46 } else {
47   // 缓存state
48   localStorage.setItem('state', state)
49   // 缓存codeVerifier
50   localStorage.setItem('codeVerifier', codeVerifier)
51   window.location.href = `${
52     import.meta.env.VITE_OAUTH_ISSUER
53   }/oauth2/authorize?response_type=code&client_id=${
54     import.meta.env.VITE_PKCE_CLIENT_ID
55   }&redirect_uri=${encodeURIComponent(
56     import.meta.env.VITE_PKCE_REDIRECT_URI
57   )}&scope=message.write%20message.read&code_challenge=${codeChallenge}&code_challenge_method=S2
58 }
59 </script>
60
61 <template>加载中...</template>
62
```

添加路由

▼

js 复制代码

```
1  {
2    path: '/PkceRedirect',
3    name: 'PkceRedirect',
4    component: () => import('../views/login/PkceRedirect.vue')
5  }
```

认证服务修改内容



重要：例如文中的就需要给客户端 `pkce-message-client` 添加一个回调地址

<http://127.0.0.1:5173/PkceRedirect>

重要：例如文中的就需要给客户端 `pkce-message-client` 添加一个回调地址

http://127.0.0.1:5173/PkceRedirect

重要：例如文中的就需要给客户端 `pkce-message-client` 添加一个回调地址

http://127.0.0.1:5173/PkceRedirect

经过以上配置授权码模式的对接就完成了，接下来就可以测试了，在首页或者需要触发登录的地方添加一个按钮，点击跳转到 `/PkceRedirect` 之后会自动引导发起授权申请流程。

最后

一直都有提到PKCE流程是授权码流程的扩展，通过这两篇文章也可以看出两种流程的授权申请流程基本是一样的，只不过PKCE将客户端密钥换成了 `code_verifier` 和 `code_challenge`，虽然稍微麻烦了些，但是安全性也提高了很多；至于移动app或者pc端应用对接的流程也是一样的，只不过是回调地址换成了URLSchema，其它都是一样的；测试的流程与授权码模式基本一致，这里就不带大家测试了，读者可自行测试，然后观察请求跳转情况。

这里贴一张获取token成功的图片

Spring Authorization Server 前后端分离示例项目

登录页面
/login

授权确认页面
/consent

设备码验证页面
/activate

验证成功页面
/activated

授权码模式
发起授权码模式的授权申请

PKCE模式
发起PKCE模式的授权申请

@稀土掘金技术社区

如果有什么问题可以在评论区指正，谢谢

代码仓库地址: [Gitee](#)
文档地址: [How-to: Authenticate using a Single Page Application with PKCE](#)

标签: Vue.js 安全 Spring Boot 话题: 日新计划更文活动

本文收录于以下专栏 1 / 2



Spring Authorization Server

Spring Authorization Server系列文章

181 订阅 · 25 篇文章

专栏目录

订阅

上一篇 Spring Authorization Server... 下一篇 Spring Authorization Server...

评论 4




平等表达，友善交流

😊 🖼️


0 / 1000 ? 发送

最热 最新

- 

用户3300833937039

全系列看完了，还有很多还需要认真学习消化，感谢楼主仔细讲解，这是截至目前网上最好的sas教程了，赞一个👍

6月前 点赞 1
- 

叹雪飞花 作者 : 谢谢 🌹



6月前 点赞 回复

...



今天吃饱了吗 后端小菜鸟 @物联网大佬

6月前 点赞 1

...



叹雪飞花 作者 : 发生甚么事了?



6月前 点赞 回复

...

目录

收起

Vue单页面项目使用授权码模式对接流程说明

Vue项目中修改内容

安装crypto-js依赖

TypeScript下额外添加@types/crypto-js依赖

编写公共方法

编写Code Verifier生成与Code Challenge的计算方法

编写获取地址栏参数方法

编写请求Token方法

在环境变量配置文件中添加配置

创建处理回调的页面PkceRedirect.vue

添加路由

认证服务修改内容

在数据库中添加对应客户端的回调地址

最后



相关推荐

Spring Authorization Server入门 (十七) Vue项目使用授权码模式对接认证服务

762阅读 · 9点赞

Spring Authorization Server入门 (十四) 联合身份认证添加微信登录

1.2k阅读 · 5点赞

Spring Authorization Server入门 (十五) 分离授权确认与设备码校验页面

1.6k阅读 · 14点赞

Spring Data Redis工具类

208阅读 · 3点赞

Spring Authorization Server入门 (六) 自定义JWT中包含的内容与资源服务jwt解析器

2.6k阅读 · 7点赞

精选内容

深度解析：Elasticsearch写入请求处理流程

公众号_醉鱼Java · 186阅读 · 2点赞

什么是线程安全？讨论的对象是什么？如何实现？

偷天神猫 · 73阅读 · 1点赞

程序员必须了解的 10个免费 Devops 工具

咸鱼运维杂谈 · 532阅读 · 5点赞

Arduino+ESP8266+华为云物联网平台实现智能开关

王二蛋呀 · 46阅读 · 0点赞

JAVA每日面经——并发编程（一）必看

阿木木AEcru · 157阅读 · 2点赞

为你推荐

Spring Authorization Server入门 (十二) 实现授权码模式使用前后端分离的登录页面

叹雪飞花 8月前 4.8k 24 65

后端 Spring ... Spring

Spring Authorization Server入门 (十三) 实现联合身份认证, 集成Github与Gitee的OAuth2.0

叹雪飞花 8月前 👁 2.3k 👍 13 💬 51

Spring Spring ... 安全

Spring Authorization Server入门 (十一) 自定义grant_type(短信认证登录)获取token

叹雪飞花 9月前 👁 2.5k 👍 15 💬 39

Spring Spring ... 安全

Spring Authorization Server入门 (七) 登录添加图形验证码

叹雪飞花 9月前 👁 2.9k 👍 18 💬 4

Spring ...

SpringBoot3.x最简集成SpringDoc-OpenApi

叹雪飞花 4月前 👁 2.3k 👍 17 💬 评论

后端 Spring ... Java

Spring Authorization Server优化篇: 添加Redis缓存支持和统一响应类

叹雪飞花 8月前 👁 1.7k 👍 6 💬 12

Spring Spring ... 安全

Spring Authorization Server入门 (十五) 分离授权确认与设备码校验页面

叹雪飞花 7月前 👁 1.6k 👍 14 💬 8

Spring ... Spring Vue.js

Spring Authorization Server入门 (十九) 基于Redis的Token、客户端信息和授权确认信...

叹雪飞花 4月前 👁 1.2k 👍 8 💬 19

Spring ... 后端 Redis

Spring Authorization Server入门 (二十) 实现二维码扫码登录

叹雪飞花 2月前 👁 935 👍 16 💬 6

Spring ... Spring Java

Spring Cloud Gateway集成SpringDoc, 集中管理微服务API

叹雪飞花 3月前 👁 767 👍 7 💬 评论

Spring ... Spring ... Java

SpringDoc枚举字段处理与SpringBoot接收枚举参数处理

叹雪飞花 4月前 👁 458 👍 5 💬 评论

后端 Spring ... Java

SpringDoc基础配置和集成OAuth2登录认证教程

叹雪飞花 4月前 👁 362 👍 4 💬 评论

后端 Spring ... Java

30 道 Vue 面试题, 内含详细讲解 (涵盖入门到精通, 自测 Vue 掌握程度)

随风而逝_风逝 4年前 👁 684k 👍 8.4k 💬 296

面试 Vue.js

Spring Authorization Server入门 (十六) Spring Cloud Gateway对接认证服务

