授权码模式

申请授权码:

http://localhost:8080/oauth/authorize?response_type=code&client_id=xwn&redirect_url=http://localhost:8080&scope=all

response_type:code 授权类型为授权码模式

client id:xwn 客户端id

redirect_url=http://localhost:8080 重定向地址

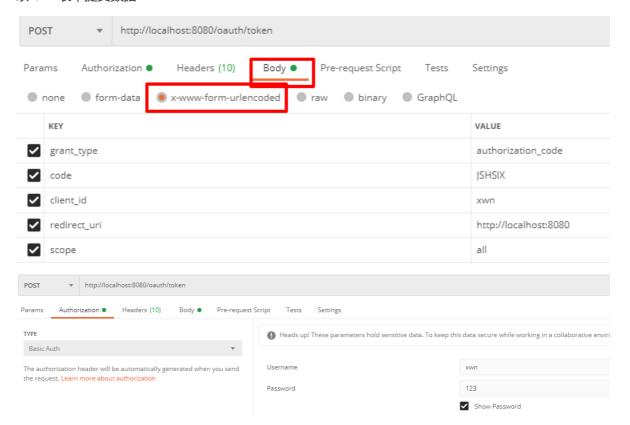
scope:all 授权范围

拿到授权码:D7lY3W

http://localhost:8080/?code=D7lY3W

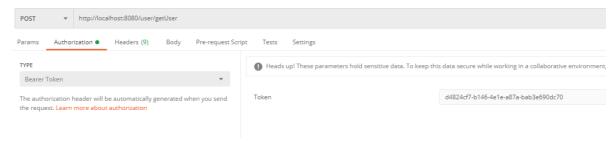
在访问:http://localhost:8080/oauth/token

以form表单提交数据



就可以获取到token,再根据token访问受保护的资源

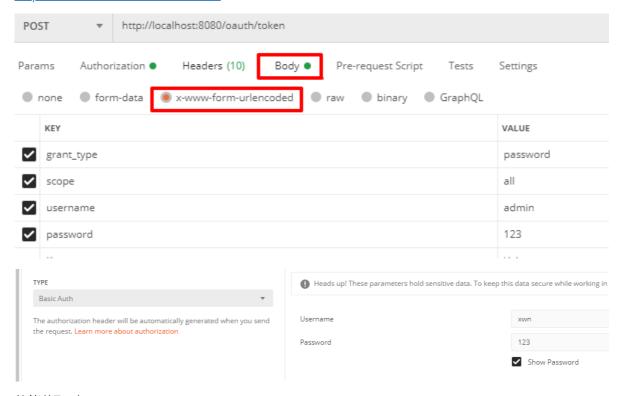
路径:http://localhost:8080/user/getUser



密码模式

直接访问:

http://localhost:8080/oauth/token



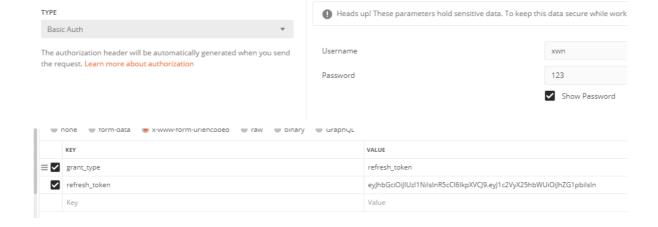
就能获取到token

```
1  {
2     "access_token": "d4824cf7-b146-4e1e-a87a-bab3e690dc70",
3     "token_type": "bearer",
4     "expires_in": 43199,
5     "scope": "all"
6  }
```

刷新令牌

直接访问:

http://localhost:8080/oauth/token



项目结构



依赖:

```
1
    <parent>
 2
            <groupId>org.springframework.boot
 3
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>2.1.4.RELEASE
 4
 5
            <relativePath/>
 6
        </parent>
        <dependencies>
8
9
            <dependency>
10
                <groupId>org.springframework.cloud
11
                <artifactId>spring-cloud-starter-oauth2</artifactId>
12
            </dependency>
            <dependency>
13
                <groupId>org.springframework.boot</groupId>
14
                <artifactId>spring-boot-starter-web</artifactId>
15
16
            </dependency>
            <!--redis-->
17
```

```
18
            <dependency>
19
                <groupId>org.springframework.boot</groupId>
20
                <artifactId>spring-boot-starter-data-redis</artifactId>
21
            </dependency>
            <!--redis存储token配置-->
22
23
            <dependency>
24
                <groupId>org.apache.commons</groupId>
25
                <artifactId>commons-pool2</artifactId>
            </dependency>
26
27
            <dependency>
                <groupId>io.jsonwebtoken
28
29
                <artifactId>jjwt</artifactId>
30
                <version>0.9.1
            </dependency>
31
32
        </dependencies>
33
34
        <dependencyManagement>
            <dependencies>
35
36
                <dependency>
37
                    <groupId>org.springframework.cloud
                    <artifactId>spring-cloud-dependencies</artifactId>
38
39
                    <version>Greenwich.SR1</version>
40
                    <type>pom</type>
41
                    <scope>import</scope>
42
                </dependency>
            </dependencies>
43
44
        </dependencyManagement>
```

授权服务器配置

```
package com.xwn.config;
 2
 3
    import com.xwn.service.DetailService;
 4
    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.beans.factory.annotation.Qualifier;
 5
 6
    import org.springframework.context.annotation.Configuration;
    import org.springframework.security.authentication.AuthenticationManager;
 7
 8
    import org.springframework.security.core.userdetails.UserDetails;
 9
    import org.springframework.security.crypto.password.PasswordEncoder;
10
    import
    org.springframework.security.oauth2.config.annotation.configurers.ClientDeta
    ilsServiceConfigurer;
11
    import
    org.springframework.security.oauth2.config.annotation.web.configuration.Auth
    orizationServerConfigurerAdapter;
12
    import
    org.springframework.security.oauth2.config.annotation.web.configuration.Enab
    leAuthorizationServer;
13
    import
    org.springframework.security.oauth2.config.annotation.web.configurers.Author
    izationServerEndpointsConfigurer;
14
    import org.springframework.security.oauth2.provider.token.TokenStore;
15
    import
    org.springframework.security.oauth2.provider.token.store.JwtAccessTokenConve
    rter;
```

```
16 import
    org.springframework.security.oauth2.provider.token.store.JwtTokenStore;
17
   /**
18
19
    * @author xwn
20
     * @date 2021/1/7 23:54
21
     */
22
23
    @Configuration
24
    @EnableAuthorizationServer
    public class AuthorizationConfig extends
25
    AuthorizationServerConfigurerAdapter {
26
27
        @Autowired
28
        private PasswordEncoder passwordEncoder;
29
30
        @Autowired
        private AuthenticationManager authenticationManager;
31
32
33
        @Autowired
        private DetailService detailService;
34
35
36
        /*@Autowired
37
        private TokenStore RedisTokenStore;*/
38
39
        @Autowired
40
        private TokenStore tokenStore;
41
42
        @Autowired
43
        private JwtAccessTokenConverter jwtAccessTokenConverter;
44
45
46
         * 密码模式需要的!*/
47
48
        @override
49
        public void configure(AuthorizationServerEndpointsConfigurer endpoints)
    throws Exception {
            endpoints.authenticationManager(authenticationManager)
50
51
                     .userDetailsService(detailService)
52
                    //把获取到的accessToken装成JWTtoken
53
                     .tokenStore(tokenStore)
54
                     .accessTokenConverter(jwtAccessTokenConverter);
55
            //redis存储token
56
            //.tokenStore(RedisTokenStore);
57
58
        }
59
60
        @override
        public void configure(ClientDetailsServiceConfigurer clients) throws
61
    Exception {
62
            clients.inMemory()
                    //客户端ID
63
                     .withClient("xwn")
64
65
                    //秘钥
66
                     .secret(passwordEncoder.encode("123"))
67
                    //重定向地址
68
                     .redirectUris("http://localhost:8080")
69
                     //令牌过期时间
```

```
70
                    .accessTokenValiditySeconds(60)
71
                    //刷新令牌过期时间
72
                    .refreshTokenValiditySeconds(86400)
73
                    //授权范围
74
                    .scopes("all")
75
                   //授权类型(authorization_code(授权码模式),password(密码模式))
76
                    .authorizedGrantTypes("authorization_code",
    "password", "refresh_token");
77
        }
78
    }
79
```

资源服务器配置

```
package com.xwn.config;
 2
 3
    import org.springframework.context.annotation.Configuration;
 4
    import
    org.springframework.security.config.annotation.web.builders.HttpSecurity;
 5
    import
    org.springframework.security.oauth2.config.annotation.web.configuration.Enab
    leResourceServer;
    import
    org.spring framework.security.oauth 2.config.annotation.web.configuration. Reso\\
    urceServerConfigurerAdapter;
 7
    /**
 8
 9
     * @author xwn
10
     * @date 2021/1/7 23:59
11
     */
12
13
    @Configuration
    @EnableResourceServer
14
    public class ResourceConfig extends ResourceServerConfigurerAdapter {
15
16
17
        @override
18
        public void configure(HttpSecurity http) throws Exception {
            http.authorizeRequests()
19
20
                    //所有请求拦截
21
                     .anyRequest()
22
                     .authenticated()
23
                     .and()
24
                     //请求路径匹配,需要token令牌才能访问
25
                     .requestMatchers()
26
                     .antMatchers("/user/**");
27
        }
    }
28
29
```

自定义登录逻辑(未连接数据库)

```
package com.xwn.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.authority.AuthorityUtils;
import org.springframework.security.core.userdetails.User;
```

```
import org.springframework.security.core.userdetails.UserDetails;
 7
    import org.springframework.security.core.userdetails.UserDetailsService;
    import
    org.springframework.security.core.userdetails.UsernameNotFoundException;
 9
    import org.springframework.security.crypto.password.PasswordEncoder;
10
    import org.springframework.stereotype.Service;
11
    /**
12
13
    * @author xwn
14
    * @date 2021/1/7 23:39
15
     */
16
17
    @service
    public class DetailService implements UserDetailsService {
18
19
20
        @Autowired
21
        private PasswordEncoder passwordEncoder;
22
23
24
        @override
25
        public UserDetails loadUserByUsername(String s) throws
    UsernameNotFoundException {
26
            return new User("admin", passwordEncoder.encode("123"),
27
28
     AuthorityUtils.commaSeparatedStringToAuthorityList("admin"));
29
        }
30
    }
31
```

security配置

```
package com.xwn.config;
 2
 3
    import org.springframework.context.annotation.Bean;
 4
    import org.springframework.context.annotation.Configuration;
    import org.springframework.security.authentication.AuthenticationManager;
 5
 6
    import
    org.springframework.security.config.annotation.web.builders.HttpSecurity;
    import
    org.springframework.security.config.annotation.web.configuration.EnableWebSe
    curity;
    import
    org.springframework.security.config.annotation.web.configuration.WebSecurity
    ConfigurerAdapter;
 9
    import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
    import\ org. spring framework. security. crypto. password. Password Encoder;
10
11
    /**
12
13
    * @author xwn
14
     * @date 2021/1/7 23:40
     */
15
16
    @Configuration
17
    @EnableWebSecurity
18
19
    public class SecurityConfig extends WebSecurityConfigurerAdapter {
```

```
21
        @override
22
        protected void configure(HttpSecurity http) throws Exception {
23
             http.authorizeRequests()
                     .antMatchers("/oauth/**","/login/**","/logout/**")
24
25
                     .permitAll()
26
                     .anyRequest().authenticated()
27
                     .and()
28
                     .csrf().disable()
29
                     .formLogin()
30
                     .permitAll();
31
        }
32
33
        @Bean
        public PasswordEncoder passwordEncoder(){
34
35
             return new BCryptPasswordEncoder();
36
        }
37
38
        //密码模式的配置
39
40
        public AuthenticationManager authenticationManager() throws Exception {
41
42
             return super.authenticationManager();
43
        }
44
    }
45
```

UserController

```
package com.xwn.controller;
 1
 2
 3
    import io.jsonwebtoken.Jwts;
    import org.springframework.security.core.Authentication;
 4
    import\ org. spring framework. we b. bind. annotation. Request Mapping;
    import org.springframework.web.bind.annotation.RestController;
 6
 7
 8
    import javax.servlet.http.HttpServletRequest;
 9
    import java.nio.charset.StandardCharsets;
10
    /**
11
     * @author xwn
12
13
     * @date 2021/1/8 0:01
14
15
16
17
    @RestController
18
    @RequestMapping("/user")
    public class UserController {
19
20
21
        @RequestMapping("/getUser")
22
        public Object getUser(Authentication authentication, HttpServletRequest
    request) {
23
            String authorization = request.getHeader("Authorization");
            String token =
    authorization.substring(authorization.lastIndexOf("bearer") + 7);
25
            return Jwts.parser()
26
                     .setSigningKey("test_key".getBytes(StandardCharsets.UTF_8))
27
                     .parseClaimsJws(token)
```

```
28 .getBody();
29 }
30 }
31
```

JwtTokenStoreConfig

```
package com.xwn.config;
 2
   import org.springframework.context.annotation.Bean;
    import org.springframework.context.annotation.Configuration;
    import org.springframework.security.oauth2.provider.token.TokenStore;
    import
    org.spring framework.security.oauth 2.provider.token.store. {\tt JwtAccessTokenConve}
    rter;
    org.springframework.security.oauth2.provider.token.store.JwtTokenStore;
 8
 9
10
    * @author xwn
11
    * @date 2021/1/8 18:25
12
13
14
    @Configuration
    public class JwtTokenStoreConfig{
15
16
17
        @Bean
18
        public TokenStore tokenStore(){
19
            return new JwtTokenStore(jwtAccessTokenConverter());
20
        }
21
22
        @Bean
23
        public JwtAccessTokenConverter jwtAccessTokenConverter(){
24
            JwtAccessTokenConverter jwtAccessTokenConverter = new
    JwtAccessTokenConverter();
25
            //jwt的密钥
26
            jwtAccessTokenConverter.setSigningKey("test_key");
27
            return jwtAccessTokenConverter;
28
        }
    }
29
30
```