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

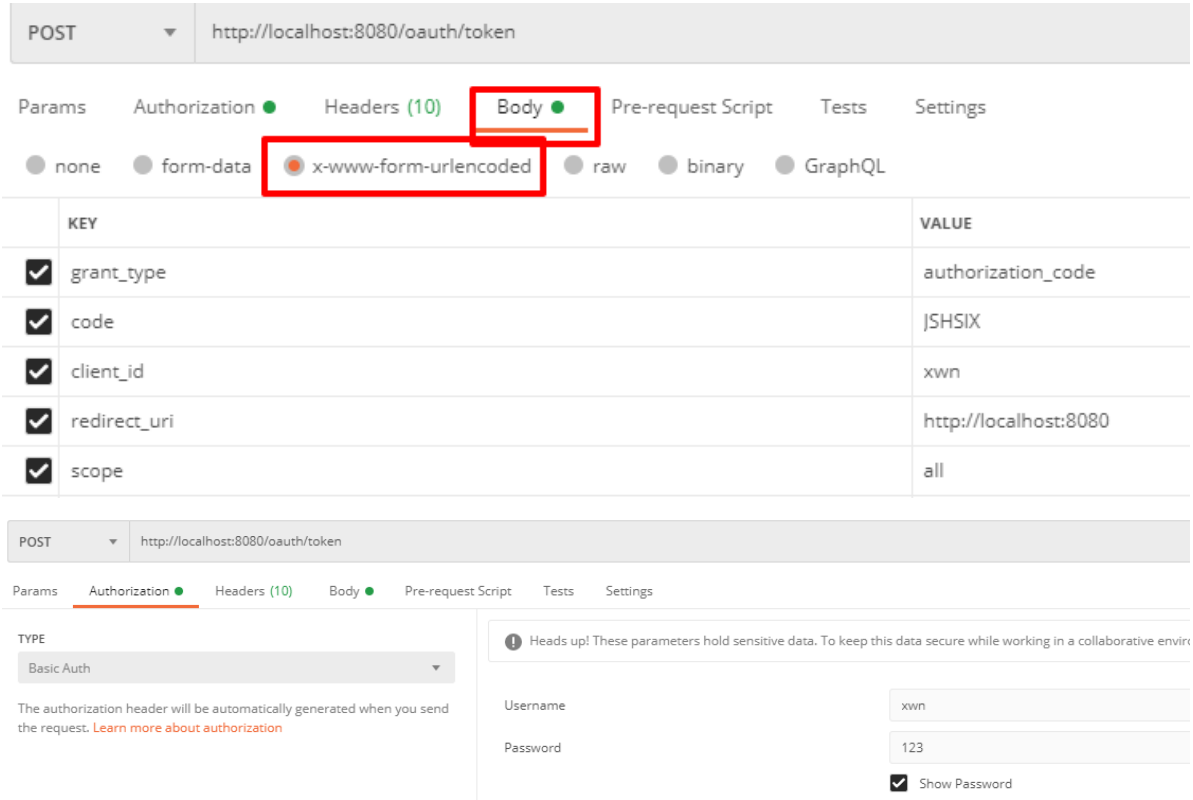
scope:all 授权范围

拿到授权码:D7lY3W

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

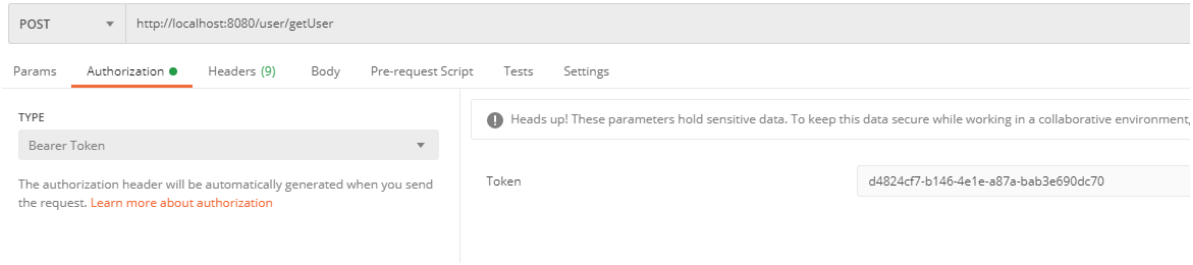
在访问:http://localhost:8080/oauth/token

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

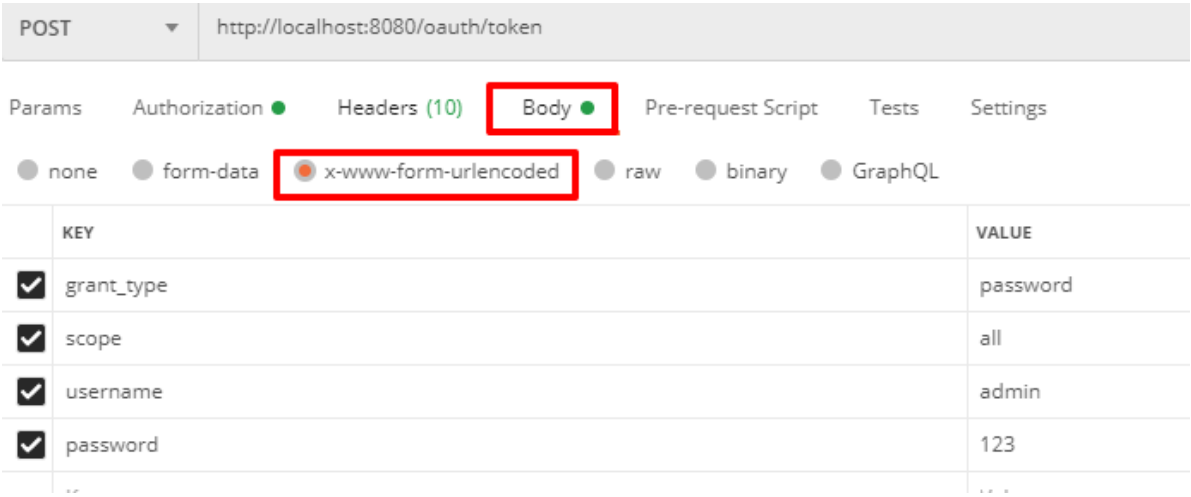
路径:http://localhost:8080/user/getUser

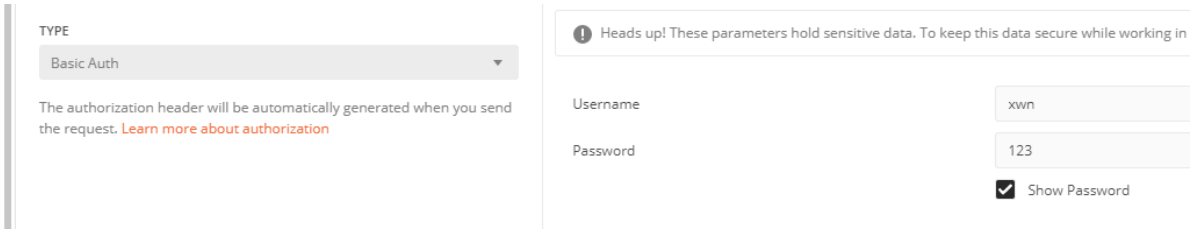


密码模式

直接访问:

http://localhost:8080/oauth/token





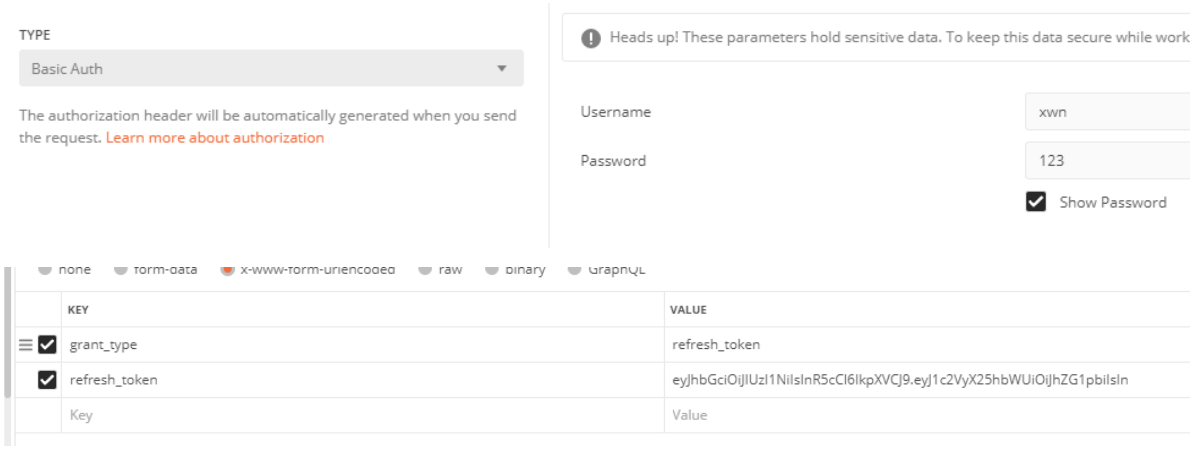
就能获取到token

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

刷新令牌

直接访问:

http://localhost:8080/oauth/token



项目结构

依赖:

```
<parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.4.RELEASE</version>
        <relativePath/>
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-oauth2</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <!--redis-->
```

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-redis</artifactId>
        </dependency>
        <!--redis存储token配置-->
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-pool2</artifactId>
        </dependency>
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt</artifactId>
            <version>0.9.1</version>
        </dependency>
    </dependencies>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>Greenwich.SR1</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
```

授权服务器配置

```
package com.xwn.config;

import com.xwn.service.DetailService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.crypto.password.PasswordEncoder;
import
org.springframework.security.oauth2.config.annotation.configurers.ClientDeta
ilsServiceConfigurer;
import
org.springframework.security.oauth2.config.annotation.web.configuration.Auth
orizationServerConfigurerAdapter;
import
org.springframework.security.oauth2.config.annotation.web.configuration.Enab
leAuthorizationServer;
import
org.springframework.security.oauth2.config.annotation.web.configurers.Author
izationServerEndpointsConfigurer;
import
org.springframework.security.oauth2.config.annotation.web.configurers.Author
izationServerSecurityConfigurer;
import org.springframework.security.oauth2.provider.token.TokenStore;
```

```java
import org.springframework.security.oauth2.provider.token.store.JwtAccessTokenConverter;
import org.springframework.security.oauth2.provider.token.store.JwtTokenStore;

/**
 * @author xwn
 * @date 2021/1/7 23:54
 */

@Configuration
@EnableAuthorizationServer
public class AuthorizationConfig extends AuthorizationServerConfigurerAdapter {

    @Autowired
    private PasswordEncoder passwordEncoder;

    @Autowired
    private AuthenticationManager authenticationManager;

    @Autowired
    private DetailService detailService;

    /*@Autowired
    private TokenStore RedisTokenStore;*/

    @Autowired
    private TokenStore tokenStore;

    @Autowired
    private JwtAccessTokenConverter jwtAccessTokenConverter;


    /*
     * 密码模式需要的!*/
    @Override
    public void configure(AuthorizationServerEndpointsConfigurer endpoints) throws Exception {
        endpoints.authenticationManager(authenticationManager)
                .userDetailsService(detailService)
                //把获取到的accessToken转成JWTtoken
                .tokenStore(tokenStore)
                .accessTokenConverter(jwtAccessTokenConverter);
        //redis存储token
        //.tokenStore(RedisTokenStore);

    }

    @Override
    public void configure(ClientDetailsServiceConfigurer clients) throws Exception {
        clients.inMemory()
                //客户端ID
                .withClient("xwn")
                //秘钥
                .secret(passwordEncoder.encode("123"))
```

```
                //重定向地址,千万不要写错(客户端的地址)
                .redirectUris("http://localhost:8081/login")
                //令牌过期时间
                .accessTokenValiditySeconds(60)
                //刷新令牌过期时间
                .refreshTokenValiditySeconds(86400)
                //授权范围
                .scopes("all")
                //自动授权,不需要手动按授权
                //.autoApprove(true)
                //授权类型(authorization_code(授权码模式),password(密码模式))
                .authorizedGrantTypes("authorization_code",
"password","refresh_token");
    }


    @Override
    public void configure(AuthorizationServerSecurityConfigurer security)
throws Exception {
        //获取秘钥必须要身份验证,单点登录必须要配置的
        security.tokenKeyAccess("isAuthenticated()");
    }
}

```

资源服务器配置

```
package com.xwn.config;

import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.oauth2.config.annotation.web.configuration.Enab
leResourceServer;
import
org.springframework.security.oauth2.config.annotation.web.configuration.Reso
urceServerConfigurerAdapter;

/**
 * @author xwn
 * @date 2021/1/7 23:59
 */

@Configuration
@EnableResourceServer
public class ResourceConfig extends ResourceServerConfigurerAdapter {

    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                //所有请求拦截
                .anyRequest()
                .authenticated()
                .and()
                //请求路径匹配,需要token令牌才能访问
                .requestMatchers()
```

```
                .antMatchers("/user/**");
    }
}

```

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;
import org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

/**
 * @author xwn
 * @date 2021/1/7 23:39
 */

@Service
public class DetailService implements UserDetailsService {

    @Autowired
    private PasswordEncoder passwordEncoder;


    @Override
    public UserDetails loadUserByUsername(String s) throws
UsernameNotFoundException {

        return new User("admin", passwordEncoder.encode("123"),

 AuthorityUtils.commaSeparatedStringToAuthorityList("admin"));
    }
}

```

security配置

```
package com.xwn.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
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.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

/**
 * @author xwn
 * @date 2021/1/7 23:40
 */

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                .antMatchers("/oauth/**","/login/**","/logout/**")
                .permitAll()
                .anyRequest().authenticated()
                .and()
                .csrf().disable()
                .formLogin()
                .permitAll();
    }

    @Bean
    public PasswordEncoder passwordEncoder(){
        return new BCryptPasswordEncoder();
    }


    //密码模式的配置
    @Bean
    public AuthenticationManager authenticationManager() throws Exception {
        return super.authenticationManager();
    }
}

```

UserController

```
package com.xwn.controller;

import io.jsonwebtoken.Jwts;
import org.springframework.security.core.Authentication;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import javax.servlet.http.HttpServletRequest;
import java.nio.charset.StandardCharsets;

/**
 * @author xwn
 * @date 2021/1/8 0:01
 */
```

```
15
16
17  @RestController
18  @RequestMapping("/user")
19  public class UserController {
20
21      @RequestMapping("/getUser")
22      public Object getUser(Authentication authentication, HttpServletRequest
    request) {
23          String authorization = request.getHeader("Authorization");
24          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

```
1   package com.xwn.config;
2
3   import org.springframework.context.annotation.Bean;
4   import org.springframework.context.annotation.Configuration;
5   import org.springframework.security.oauth2.provider.token.TokenStore;
6   import
    org.springframework.security.oauth2.provider.token.store.JwtAccessTokenConve
    rter;
7   import
    org.springframework.security.oauth2.provider.token.store.JwtTokenStore;
8
9   /**
10   * @author xwn
11   * @date 2021/1/8 18:25
12   */
13
14  @Configuration
15  public class JwtTokenStoreConfig{
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
```

客户端配置

项目结构

依赖

```
<parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.4.RELEASE</version>
        <relativePath/>
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-oauth2</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt</artifactId>
            <version>0.9.1</version>
        </dependency>
    </dependencies>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>Greenwich.SR1</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
```

application.yml

```yaml
server:
  port: 8081
  servlet:
    session:
      cookie:
        name: OAUTH2-CLIENT-SESSIONID01 #修改cookie的key,防止cookie冲突,冲突会导致登录验证不通过


#授权服务器地址
oauth2-server-url: http://localhost:8080

#授权服务器相关配置
security:
  oauth2:
    client:
      client-id: xwn #客户端id
      client-secret: 123 #客户端秘钥
      user-authorization-uri: ${oauth2-server-url}/oauth/authorize #获取授权码地址
      access-token-uri: ${oauth2-server-url}/oauth/token #获取access_key的地址
    resource:
      jwt:
        key-uri: ${oauth2-server-url}/oauth/token_key #获取jwt令牌的地址
```

启动类

```java
package com.xwn;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.security.oauth2.client.EnableOAuth2Sso;

/**
 * @author xwn
 * @date 2021/1/9 16:00
 */

@SpringBootApplication
//单点登录注解
@EnableOAuth2Sso
public class ClientApplication {
    public static void main(String[] args) {
        SpringApplication.run(ClientApplication.class,args);
    }
}

```

UserController

```java
package com.xwn.controller;

import org.springframework.security.core.Authentication;
import org.springframework.web.bind.annotation.RequestMapping;
```

```java
import org.springframework.web.bind.annotation.RestController;

/**
 * @author xwn
 * @date 2021/1/9 16:04
 */


@RestController
@RequestMapping("/user")
public class UserController {


    @RequestMapping("/getUserInfo")
    public Object getUserInfo(Authentication authentication){
        return authentication;
    }
}

```

访问http://localhost:8081/user/getUserInfo

自动根据配置文件找到授权服务器