Spring Cloud Security：Oauth2使用入门

原创 梦想de星空 macrozheng 2019-11-04 08:32

收录于合集
#Spring Cloud学习教程

26个

> Spring Cloud Security 为构建安全的SpringBoot应用提供了一系列解决方案，结合Oauth2可以实现单点登录、令牌中继、令牌交换等功能，本文将对其结合Oauth2入门使用进行详细介绍。

OAuth2 简介

OAuth 2.0是用于授权的行业标准协议。OAuth 2.0为简化客户端开发提供了特定的授权流，包括Web应用、桌面应用、移动端应用等。

OAuth2 相关名词解释

- Resource owner（资源拥有者）：拥有该资源的最终用户，他有访问资源的账号密码；
- Resource server（资源服务器）：拥有受保护资源的服务器，如果请求包含正确的访问令牌，可以访问资源；
- Client（客户端）：访问资源的客户端，会使用访问令牌去获取资源服务器的资源，可以是浏览器、移动设备或者服务器；
- Authorization server（认证服务器）：用于认证用户的服务器，如果客户端认证通过，发放访问资源服务器的令牌。

四种授权模式

- Authorization Code（授权码模式）：正宗的OAuth2的授权模式，客户端先将用户导向认证服务器，登录后获取授权码，然后进行授权，最后根据授权码获取访问令牌；
- Implicit（简化模式）：和授权码模式相比，取消了获取授权码的过程，直接获取访问令牌；

- Resource Owner Password Credentials（密码模式）：客户端直接向用户获取用户名和密码，之后向认证服务器获取访问令牌；
- Client Credentials（客户端模式）：客户端直接通过客户端认证（比如client_id和client_secret）从认证服务器获取访问令牌。

两种常用的授权模式

授权码模式

- (A)客户端将用户导向认证服务器；
- (B)用户在认证服务器进行登录并授权；
- (C)认证服务器返回授权码给客户端；
- (D)客户端通过授权码和跳转地址向认证服务器获取访问令牌；
- (E)认证服务器发放访问令牌（有需要带上刷新令牌）。

密码模式

- (A)客户端从用户获取用户名和密码；
- (B)客户端通过用户的用户名和密码访问认证服务器；
- (C)认证服务器返回访问令牌（有需要带上刷新令牌）。

Oauth2的使用

创建oauth2-server模块

> 这里我们创建一个oauth2-server模块作为认证服务器来使用。

- 在pom.xml中添加相关依赖：

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-oauth2</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

- 在application.yml中进行配置：

```
server:
  port:9401
spring:
  application:
    name:oauth2-service
```

- 添加UserService实现UserDetailsService接口，用于加载用户信息：

```
/**
 * Created by macro on 2019/9/30.
 */
@Service
publicclass UserService implements UserDetailsService {
    private List<User> userList;
    @Autowired
    private PasswordEncoder passwordEncoder;

    @PostConstruct
    public void initData() {
        String password = passwordEncoder.encode("123456");
        userList = new ArrayList<>();
        userList.add(new User("macro", password, AuthorityUtils.commaSeparatedStringToAuthorityLi
        userList.add(new User("andy", password, AuthorityUtils.commaSeparatedStringToAuthorityLis
        userList.add(new User("mark", password, AuthorityUtils.commaSeparatedStringToAuthorityLis
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        List<User> findUserList = userList.stream().filter(user -> user.getUsername().equals(user
        if (!CollectionUtils.isEmpty(findUserList)) {
            return findUserList.get(0);
        } else {
            thrownew UsernameNotFoundException("用户名或密码错误");
        }
    }
}
```

- 添加认证服务器配置，使用@EnableAuthorizationServer注解开启：

```
/**
 * 认证服务器配置
 * Created by macro on 2019/9/30.
 */
@Configuration
@EnableAuthorizationServer
publicclass AuthorizationServerConfig extends AuthorizationServerConfigurerAdapter {

    @Autowired
    private PasswordEncoder passwordEncoder;

    @Autowired
    private AuthenticationManager authenticationManager;

    @Autowired
    private UserService userService;

    /**
     * 使用密码模式需要配置
     */
    @Override
    public void configure(AuthorizationServerEndpointsConfigurer endpoints) {
        endpoints.authenticationManager(authenticationManager)
                .userDetailsService(userService);
    }

    @Override
    public void configure(ClientDetailsServiceConfigurer clients) throws Exception {
        clients.inMemory()
                .withClient("admin")//配置client_id
                .secret(passwordEncoder.encode("admin123456"))//配置client_secret
                .accessTokenValiditySeconds(3600)//配置访问token的有效期
                .refreshTokenValiditySeconds(864000)//配置刷新token的有效期
                .redirectUris("http://www.baidu.com")//配置redirect_uri，用于授权成功后跳转
                .scopes("all")//配置申请的权限范围
                .authorizedGrantTypes("authorization_code","password");//配置grant_type，表示授权类
    }
}
```

- 添加资源服务器配置，使用@EnableResourceServer注解开启：

```
/**
 * 资源服务器配置
 * Created by macro on 2019/9/30.
 */
@Configuration
@EnableResourceServer
publicclass ResourceServerConfig extends ResourceServerConfigurerAdapter {

    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                .anyRequest()
                .authenticated()
                .and()
                .requestMatchers()
                .antMatchers("/user/**");//配置需要保护的资源路径
    }
}
```

- 添加SpringSecurity配置，允许认证相关路径的访问及表单登录：

```
/**
 * SpringSecurity配置
 * Created by macro on 2019/10/8.
 */
@Configuration
@EnableWebSecurity
publicclass SecurityConfig extends WebSecurityConfigurerAdapter {

    @Bean
    public PasswordEncoder passwordEncoder() {
        returnnew BCryptPasswordEncoder();
    }

    @Bean
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        returnsuper.authenticationManagerBean();
    }

    @Override
```

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

- 添加需要登录的接口用于测试：

```
/**
 * Created by macro on 2019/9/30.
 */
@RestController
@RequestMapping("/user")
publicclass UserController {
    @GetMapping("/getCurrentUser")
    public Object getCurrentUser(Authentication authentication) {
        return authentication.getPrincipal();
    }
}
```

授权码模式使用

- 启动oauth2-server服务；
- 在浏览器访问该地址进行登录授权：http://localhost:9401/oauth/authorize?response_type=code&client_id=admin&redirect_uri=http://www.baidu.com&scope=all&state=normal
- 输入账号密码进行登录操作：

- 登录后进行授权操作：

- 之后会浏览器会带着授权码跳转到我们指定的路径：

```
https://www.baidu.com/?code=eTsADY&state=normal
```

- 使用授权码请求该地址获取访问令牌：http://localhost:9401/oauth/token
- 使用Basic认证通过client_id和client_secret构造一个Authorization头信息；

- 在body中添加以下参数信息，通过POST请求获取访问令牌；

- 在请求头中添加访问令牌，访问需要登录认证的接口进行测试，发现已经可以成功访问：http://localhost:9401/user/getCurrentUser

密码模式使用

- 使用密码请求该地址获取访问令牌：http://localhost:9401/oauth/token

- 使用Basic认证通过client_id和client_secret构造一个Authorization头信息；

- 在body中添加以下参数信息，通过POST请求获取访问令牌；

使用到的模块

```
springcloud-learning
└── oauth2-server -- oauth2认证测试服务
```

项目源码地址

https://github.com/macrozheng/springcloud-learning

推荐阅读

- 这样讲**API**网关，你应该能明白了吧！
- 使用策略**+**工厂模式彻底干掉代码中的**if else**！
- 后端程序员必备：**Mysql**数据库相关流程图**/**原理图
- 【真实生产案例】消息中间件如何处理消费失败的消息？
- 我的**Github**开源项目，从**0**到**20000 Star**！
- **Spring Boot Admin**：微服务应用监控
- **Spring Cloud Gateway**：新一代**API**网关服务
- **Spring Cloud Consul**：服务治理与配置中心
- **Spring Cloud Sleuth**：分布式请求链路跟踪
- **Spring Cloud Bus**：消息总线
- **Spring Cloud Config**：外部集中化配置管理
- **Spring Cloud Zuul**：**API**网关服务

欢迎关注，点个在看

阅读原文

喜欢此内容的人还喜欢

项目中到底该不该用Lombok?

macrozheng