链滴

MySQL+SpringBoot+OAuth2.0 配置

作者：someone45057
原文链接：https://ld246.com/article/1555388497549
来源网站：链滴

OAuth2.0规范

是微服务以及前后分离的趋势下产生的规范，用于客户端请求服务端资源，这里的客户端可以是一个端项目或者是另一个服务端。简单来说就是端到端的请求。

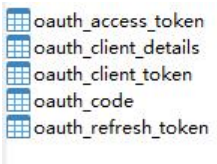
下面简单讲一下使用步骤，我这里的SpringBoot的版本是当前最新版：2.1.4

MySQL配置：

- MySQL建表，建表语句github上有，参见：

https://gist.github.com/leolin310148/3b2cb7d83ba0ec9e1d58

完成之后，数据库里应该有这几张表：



- 在oauth_client_details里添加一条数据：

client_id: client_1
resource_ids: demo
client_secret: 123456
scope: select
authorized_grant_types: client_credentials
web_server_redirect_uri: null
authorities: client
access_token_validity: null
refersh_token_validity: null
additional_infomation: null,
autoapprove: 1

application.propertites配置：

```
spring.datasource.url=jdbc:mysql://你的数据库ip:你的数据库端口?serverTimezone=UTC&useUnicode=true&characterEncoding=utf-8&useSSL=true
spring.datasource.username=你的数据库用户名
spring.datasource.password=你的数据密码
server.port=你的项目监听端口
```

maven配置：

```
<!-- SpringBoot基本配置 -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<!-- devtools 开发热部署 -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
```

```
    <scope>runtime</scope>
</dependency>
<!-- SpringBoot 测试支持 -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
<!-- 这两个是SpringBoot+OAuth2.0的核心配置，我这里的SpringBoot版本是2.1.4所以下面的两
版本需要对应一下 -->
<dependency>
    <groupId>org.springframework.security.oauth.boot</groupId>
    <artifactId>spring-security-oauth2-autoconfigure</artifactId>
    <version>2.1.4.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.security.oauth</groupId>
    <artifactId>spring-security-oauth2</artifactId>
    <version>2.3.5.RELEASE</version>
</dependency>
```

这里的OAuth2.0核心配置需要注意，根据2.1.4版本的官方文档：

https://docs.spring.io/spring-boot/docs/2.1.4.RELEASE/reference/htmlsingle/#_authorization_erver

Currently, Spring Security does not provide support for implementing an OAuth 2.0 Authorization Server. However, this functionality is available from theSpring Security OAuthproject, whic will eventually be superseded by Spring Security completely. Until then, you can use thesprin-security-oauth2-autoconfiguremodule to easily set up an OAuth 2.0 authorization server; see itsdocumentationfor instructions.

意思就是Spring Security不再提供OAuth2.0认证服务器的实现，转而由Spring Security OAuth项实现。

SpringBoot注解配置

启动类：

```
@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

ResourseServer资源服务器配置，新建一个RSConfig类，内容如下：

```
/*
 * 资源服务器
 * */
@Configuration
@EnableResourceServer
```

```
public class RSConfig extends ResourceServerConfigurerAdapter {
    @Override
    public void configure(HttpSecurity http) throws Exception {
        http
            .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.IF_REQUIRED)
            .and()
            .requestMatchers().anyRequest()
            .and()
            .anonymous()
            .and()
            .authorizeRequests()
            .antMatchers("/demo/**").authenticated();
    }
    @Override
    public void configure(ResourceServerSecurityConfigurer resources) throws Exception {
        resources.resourceId("demo");
    }
}
```

这里就是配置/demo/下的任何请求都需要认证，并且将数据库的那一条记录的resource_ids里的dem注册为资源

AuthorizationServer认证服务器配置，新建一个ASConfig类，内容如下：

```
/*
 * 认证服务器
 * */
@Configuration
@EnableAuthorizationServer
public class ASConfig extends AuthorizationServerConfigurerAdapter {
    @Autowired
    private DataSource dataSource;
    @Override
    public void configure(ClientDetailsServiceConfigurer clients) throws Exception {
        clients.jdbc(dataSource);
    }
    @Override
    public void configure(AuthorizationServerEndpointsConfigurer endpoints) throws Exception
{
        endpoints
                .tokenStore(new JdbcTokenStore(dataSource));
    }
    @Override
    public void configure(AuthorizationServerSecurityConfigurer oauthServer) throws Exception
{
        //允许表单认证
        oauthServer.allowFormAuthenticationForClients();
    }
}
```

这里的DataSource就是application.propertities里配置的，SpringBoot自动注入了，不用我们关心。

SpringSecurity配置，新建一个SecurityConfig，写入以下内容：

```
/*
 * Spring Security配置
 * */
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    @Configuration
    @EnableWebSecurity
    public class SecurityConfiguration extends WebSecurityConfigurerAdapter {
        @Override
        protected void configure(HttpSecurity http) throws Exception {
            http
                .requestMatchers().anyRequest()
                .and()
                .authorizeRequests()
                .antMatchers("/oauth/*").permitAll();
        }
    }
}
```

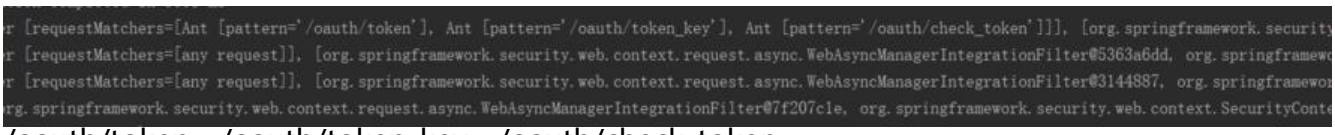
这里其实可以理解为SpringSecurity的拦截发生在OAuth之前，所以需要允许所有的/oauth/下的请
。

Controller，就是受保护的资源，新建一个DemoController，内容如下：

```
@RestController
@RequestMapping("/demo")
public class DemoController {
    @RequestMapping("/hello")
    public String hello(){
        return "hello~~~~!";
    }
}
```

调试结果

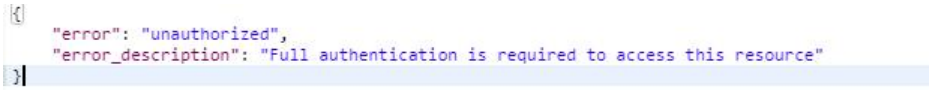
- 启动项目，可以看到如下启动信息：



/oauth/token，/oauth/token_key，/oauth/check_token

这三个接口是SpringBoot自己加上的

- 用http工具测试请求
- 先直接访问：localhost:你的端口/demo/hello，响应如下：



- 下面再请求token：http://localhost:你的端口/oauth/token

请求过程跟参数请参考：

https://oauth.net/2/

请求结果：

Token Name	Token Name
Access Token	d0de054d-08b3-4325-a254-737deed5af4b
Token Type	bearer
expires_in	43199
scope	select

- 再携带得到的token请求/demo/hello，响应如下：

```
1 hello
```

到此，整个OAuth2.0认证过程完成！
