oauth2.0

https://blog.csdn.net/u013310119/article/details/94613332

https://segmentfault.com/a/1190000016583573 短信

https://github.com/zhangwei900808/awbeci-ssb 源码

6 .OAuth2.0

6.1 OAuth2.0介绍

OAuth（开放授权）是一个开放标准，允许用户授权第三方应用访问他们存储在另外的服务提供者上的信息，而不需要将用户名和密码提供给第三方应用或分享他们数据的所有内容。OAuth2.0是OAuth协议的延续版本，但不向后兼容OAuth 1.0即完全废止了OAuth1.0。很多大公司如Google，Yahoo，Microsoft等都提供了OAUTH认证服务，这些都足以说明OAUTH标准逐渐成为开放资源授权的标准。

Oauth协议目前发展到2.0版本，1.0版本过于复杂，2.0版本已得到广泛应用。

参考：https://baike.baidu.com/item/oAuth/7153134?fr=aladdin

Oauth协议：https://tools.ietf.org/html/rfc6749

下边分析一个Oauth2认证的例子，通过例子去理解OAuth2.0协议的认证流程，本例子是黑马程序员网站使用微信认证的过程，这个过程的简要描述如下：

用户借助微信认证登录黑马程序员网站，用户就不用单独在黑马程序员注册用户，怎么样算认证成功吗？黑马程序员网站需要成功从微信获取用户的身份信息则认为用户认证成功，那如何从微信获取用户的身份信息？用户信息的拥有者是用户本人，微信需要经过用户的同意方可为黑马程序员网站生成令牌，黑马程序员网站拿此令牌方可从微信获取用户的信息。

方法授权

4.7.3.方法授权

现在我们已经掌握了使用如何使用`http.authorizeRequests()`对web资源进行授权保护，从Spring Security2.0版本开始，它支持服务层方法的安全性的支持。本节学习@PreAuthorize,@PostAuthorize,@Secured三类注解。

我们可以在任何`@Configuration`实例上使用`@EnableGlobalMethodSecurity`注释来启用基于注解的安全性。

以下内容将启用Spring Security的`@Secured`注释。

```
@EnableGlobalMethodSecurity(securedEnabled = true)
public class MethodSecurityConfig {// ...}
```

然后向方法（在类或接口上）添加注解就会限制对该方法的访问。 Spring Security的原生注释支持为该方法定义了一组属性。 这些将被传递给AccessDecisionManager以供它作出实际的决定：

```
public interface BankService {

@Secured("IS_AUTHENTICATED_ANONYMOUSLY")
public Account readAccount(Long id);
```

```
public Account post(Account account, double amount);
}
```

以上配置标明readAccount、findAccounts方法可匿名访问，底层使用WebExpressionVoter投票器，可从AffirmativeBased第23行代码跟踪。。

post方法需要有TELLER角色才能访问，底层使用RoleVoter投票器。

使用如下代码可启用prePost注解的支持

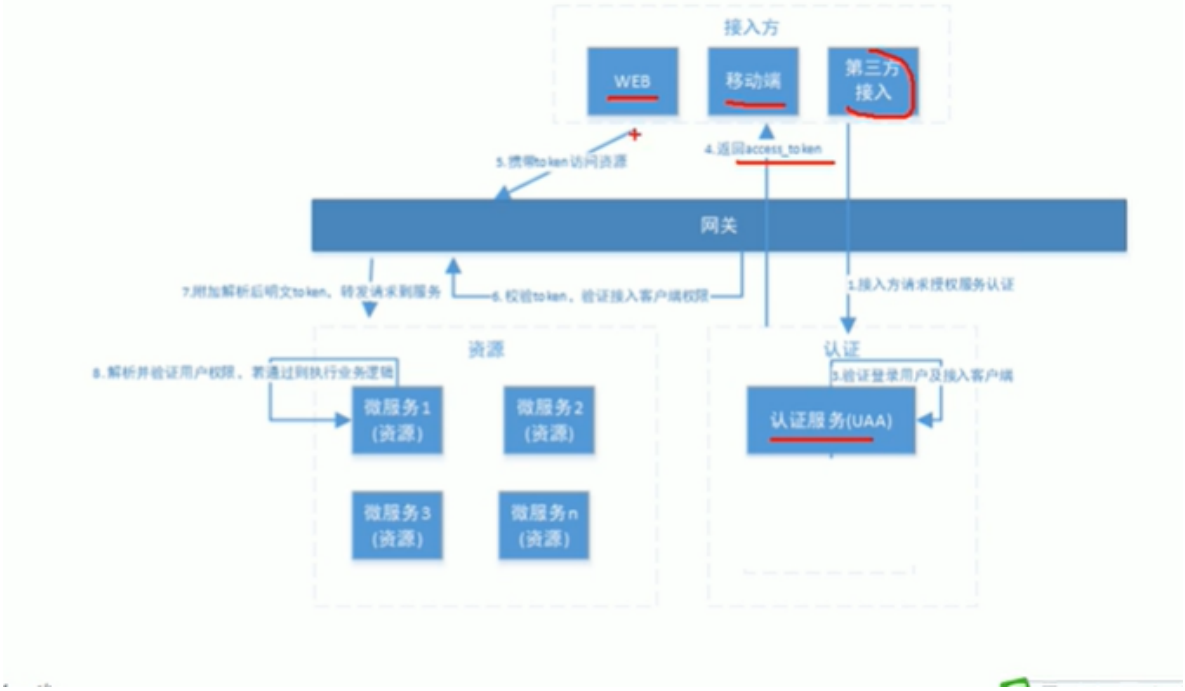
```
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class MethodSecurityConfig {
// ...
}
```

相应Java代码如下：

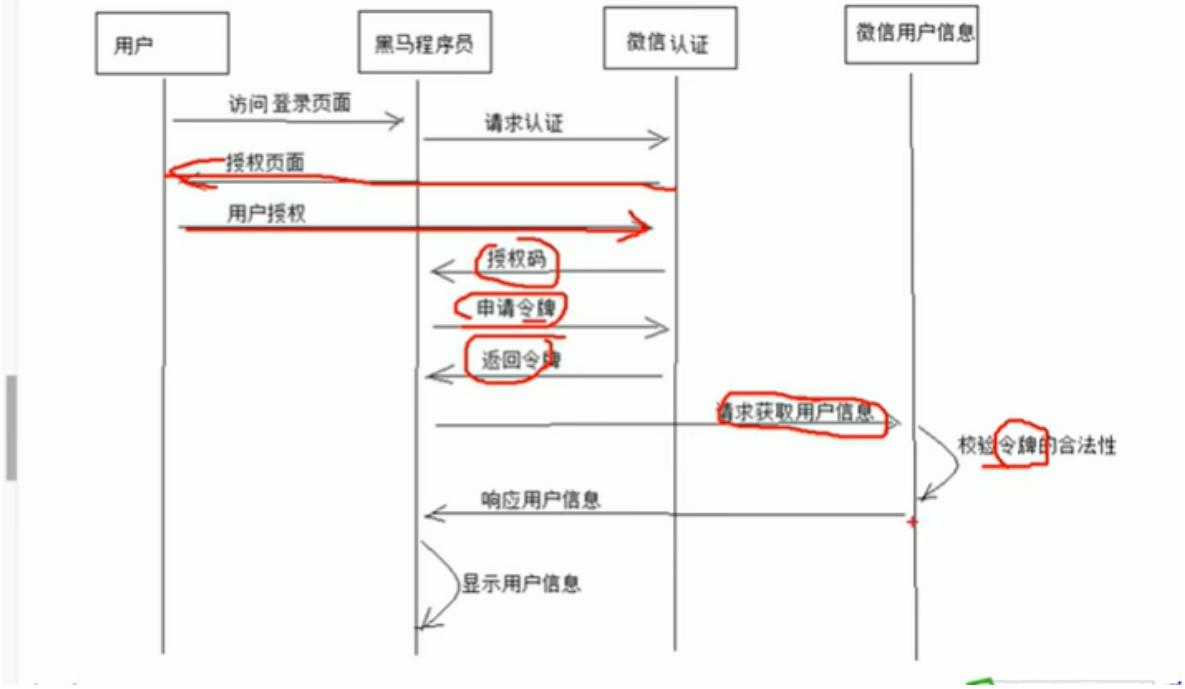
```
public interface BankService {

@PreAuthorize("isAnonymous()")
public Account readAccount(Long id);
```

分布式系统认证技术方案见下图：



以上认证授权详细的执行流程如下：



6.2 Spring Cloud Security OAuth2

6.2.1 环境介绍

Spring-Security-OAuth2是对OAuth2的一种实现，并且跟我们之前学习的Spring Security相辅相成，与Spring Cloud体系的集成也非常便利，接下来，我们需要对它进行学习，最终使用它来实现我们设计的分布式认证授权解决方案。

OAuth2.0的服务提供方涵盖两个服务，即授权服务 (Authorization Server，也叫认证服务) 和资源服务 (Resource Server)，使用 Spring Security OAuth2 的时候你可以选择把它们在同一个应用程序中实现，也可以选择建立使用同一个授权服务的多个资源服务。

授权服务 (Authorization Server) 应包含对接入端以及登入用户的合法性进行验证并颁发token等功能，对令牌的请求端点由 Spring MVC 控制器进行实现，下面是配置一个认证服务必须要实现的endpoints：

- **AuthorizationEndpoint** 服务于认证请求。默认 URL： `/oauth/authorize`。
- **TokenEndpoint** 服务于访问令牌的请求。默认 URL： `/oauth/token`。

资源服务 (Resource Server)，应包含对资源的保护功能，对非法请求进行拦截，对请求中token进行解析鉴权等，下面的过滤器用于实现 OAuth 2.0 资源服务：

- OAuth2AuthenticationProcessingFilter用来对请求给出的身份令牌解析鉴权。

```
//配置客户端详细信息服务
@Override
public void configure(ClientDetailsServiceConfigurer clients)
        throws Exception {
    //暂时使用内存方式
    clients.inMemory()// 使用in-memory存储
            .withClient( clientId: "c1")// client_id
            .secret(new BCryptPasswordEncoder().encode( rawPassword: "secret"))//客户端密钥
            .resourceIds("res1")//资源列表
            .authorizedGrantTypes("authorization_code", "password","client_credentials","implicit","re
            .scopes("all")// 允许的授权范围
            .autoApprove(false)//false 跳转到授权页面
            //加上验证回调地址
            .redirectUris("http://www.baidu.com");
}
```

```
@Bean
public AuthorizationServerTokenServices tokenService() {
    DefaultTokenServices service=new DefaultTokenServices();
    service.setClientDetailsService(clientDetailsService);//客户端信息服务
    service.setSupportRefreshToken(true);//是否产生刷新令牌
    service.setTokenStore(tokenStore);//令牌存储策略
    service.setAccessTokenValiditySeconds(7200); // 令牌默认有效期2小时
    service.setRefreshTokenValiditySeconds(259200); // 刷新令牌默认有效期3天
    return service;
}
```

```
http
                .formLogin()
                .loginPage("/auth/login")
                .loginProcessingUrl("/auth/form")
                .successHandler(authenticationSuccessHandler)
                .failureHandler(authenticationFailureHandler);
```

```
package com.oauth2.handler;

import org.codehaus.jackson.map.ObjectMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.oauth2.common.OAuth2AccessToken;
import
org.springframework.security.oauth2.common.exceptions.UnapprovedClientAuthentica
tionException;
import org.springframework.security.oauth2.provider.*;
import
org.springframework.security.oauth2.provider.token.AuthorizationServerTokenServi
ces;
import
org.springframework.security.web.authentication.SavedRequestAwareAuthenticationS
uccessHandler;
import org.springframework.stereotype.Component;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;

@Component
public class MyAuthenticationSuccessHandler extends
SavedRequestAwareAuthenticationSuccessHandler {

    @Autowired
    private ClientDetailsService clientDetailsService;
    private ObjectMapper objectMapper = new ObjectMapper();

    @Autowired
    private AuthorizationServerTokenServices authorizationServerTokenServices;

    @Override
    public void onAuthenticationSuccess(HttpServletRequest request,
HttpServletResponse response, Authentication authentication) throws
ServletException, IOException {
//        String header = request.getHeader("Authorization");
//        String name = authentication.getName();
//        if (header == null || header.startsWith("Basic ")) {
```

```
//            throw new UnapprovedClientAuthenticationException("请求头没有client
信息");
//        }
//        String[] tokens = extract(header);
        String clientId = "clientid";//tokens[0];
        String secret = "secret";//tokens[1];
        ClientDetails clientDetails =
clientDetailsService.loadClientByClientId(clientId);
        if (clientDetails == null) {
            throw new UnapprovedClientAuthenticationException("clientId信息不存
在");
        }
        if (!secret.equals(clientDetails.getClientSecret())) {
            throw new UnapprovedClientAuthenticationException("getClientSecret
不匹配");
        }
        Map<String, String> requestParameters = new HashMap<>();
        TokenRequest tokenRequest = new TokenRequest(requestParameters,
clientId, clientDetails.getScope(), "custom");
        OAuth2Request oAuth2Request =
tokenRequest.createOAuth2Request(clientDetails);
        OAuth2Authentication oAuth2Authentication = new
OAuth2Authentication(oAuth2Request, authentication);
        OAuth2AccessToken accessToken =
authorizationServerTokenServices.createAccessToken(oAuth2Authentication);
        response.setContentType("application/json;charset=UTf-8");

response.getWriter().write(objectMapper.writeValueAsString(accessToken));
    }
    private String[] extract(String header) {
        String token = null;
        try {
            byte[] base64Token = header.substring(6).getBytes("UTF-8");
            byte[] decodeToken = Base64.getDecoder().decode(base64Token);
            token = new String(decodeToken, "UTF-8");
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        int delim = token.indexOf(":");
        if (delim == -1) {
            throw new UnapprovedClientAuthenticationException("请求头没有client信
息");
        }
        return new String[] {token.substring(0, delim),
token.substring(delim+1)};



    }
}
```

```
http://localhost:9090/oauth/authorize?
response_type=code&client_id=clientid&secret=secret&redirect_uri=http://www.baid
u.com
```