地址： https://docs.spring.io/spring-security/site/docs/5.0.10.RELEASE/reference/htmlsingle/#overall-architecture

9.1 运行环境

spring security 3.0 需要Java 5 以上，不需要其他别的文件配置，即使是在ejb或者其他servlet容器中.

9.2 核心组件

9.2.1 SecurityContextHolder， SecurityContext and Authentication Objects

SecurityContextHolder： 使用ThreadLocal来存储主要的细节,是安全应用上下文的主要的类.

SecurityContextHolder.getContext().getAuthentication().getPrincipal();

SecurityContext 接口 : getContext()，返回 SecurityContextImpl 实例，有Authentication 属性.

Authentication 接口： getAuthentication()返回Authentication实例. 该实例中存放者Object principal; 大部分时候 实际就是UserDetails， 可以直接强转

UserDetails 接口 : .getPrincipal() 将返回该UserDetails 实例 有 User SocialUserDetails 等实现类 存放着具体的用户信息 username 和 password 还有4个boolean方法 后面详细说.

9.2.2 UserDetailsService

UserDetails loadUserByUsername(String username) throws UsernameNotFoundException;只有这一个方法， 就是根据username来自己操作数据库来查询相应的password，然后返回一个封装的UserDetails实例. 这个实例会返回被AuthenticationManager相应的实例进行验证.

9.2.4 总结

SecurityContextHolder （类）： 提供访问SecurityContext

SecurityContext （接口）： 存放 Authentication 和 特殊的 request 安全信息

Authentication （接口）： 认证信息的接口。如 UsernamePasswordAuthenticationToken， OAuth2Authentication 等等 相当于令牌的实例的抽象

GrantedAuthority（接口）：相对于userDetails的权限

UserDetails（接口）：组成 Authentication 实例的对象，存放 UserDetailsService 查询到的信息

UserDetailsService（接口）：创建一个UserDetails

9.3 Authentication

9.3.1 认证流程

1.根据 username 和 password 组装成UsernamePasswordAuthenticationToken对象（实际就是Authentication接口的实现）

2.将上述令牌对象交给AuthenticationManager 的实例对象进行校验

3.AuthenticationManager实例会在验证成功后，填充Authentication对象实例

4.然后用SecurityContextHolder.getContext().setAuthentication(...).将实例放到SecurityContext上下文中

9.4 Authentication in a Web Application

9.4.1 ExceptionTranslationFilter： 检测所有的spring security抛出的异常

9.4.2 AuthenticationEntryPoint : 对未认证的请求，返回的response 信息包装.

9.4.3 认证机制

1.用户点击连接，请求服务器

2.请求被服务器保护资源

3.用户未认证，服务器让浏览器去认证 或者 返回信息

4.用户去向认证页面 或者 收到返回信息

5.用户向服务器发送一个请求，并且包含具体的认证信息（eg:一个form）

6.服务器决定是否认证通过，如果是，则进行第七步. 没有通过认证，将让用户从新认证，就是跳到第三步.

7.认证成功后，访问受保护资源，如果权限够，则成功；权限不够，则返回403.

summary:服务器收集用户发的信息组装成Authentication 交给AuthenticationManager来进行处理（认证），认证成功则进到第七步；如果，AuthenticationManager拒绝了这个认证信息，则，到第三步.

9.4.4 2018/11/29 22:00 明天再看

9.4.4 存储SecurityContext在请求之间

SecurityContextPersistenceFilter 会存储SecurityContext ， 用 HttpSession 在不同的请求之间存放. 请求结束后，清除。 不需要我们干预.

9.5.2 AbstractSecurityInterceptor

工作流程：

1.收集该请求的相关配置属性 configuration attributes （比如：<intercept-url pattern='/secure/**' access='ROLE_A,ROLE_B'/>）即，/secure/**下的url需要ROLE_A,ROLE_B权限才能访问.

2 . 提交 当前的安全对象，Authentication 和配置属性 给 AccessDecisionManager 去进行验证.

3.Authentication 可以被替换

4.安全对象可以被加工（如果授权成功）

5.一旦调用验证的方法返回，就调用AfterInvocationManager （如果配置），如果抛出异常，就不调用.