Spring Security

简介：

Spring Security 是Spring社区的一个顶级项目，也是SpringBoot官方推荐使用的Security框架。提供了很多高级特性以满足复杂场景下的安全需求。虽然共更能强大，但配置并不复杂（扯淡），得益于官方详尽的文档（完全看不懂文档）。在使用层面，Spring Security 提供了多种方式进行业务集成，包括注解，Servlet API，JSP Tag，系统API等。

1. 核心概念：

Principle(User)，Authority(Role)和Permission是Spring Security的3个核心概念。

跟通常理解上的Role和Permission之间一对多的关系不同，在Spring Security中，Authority和Permission是两个完全独立的概念，两者没有必然联系，但是可以通过配置进行关联。

应用级别的安全主要分为验证（authentication）和授权（authorization）两部分。这也是Spring Security主要需要处理的两个部分：

认证过程称之为Authentication（验证）,指的是系统使用者信息（principal）的过程，即用户或设备登录操作。Authorization指的是判断某个principal在我们的应用是否允许执行某个操作。在进行授权判断之前，要求该principal已经通过Authentication的认证。

场景：

1. 一个用户被提示使用用户名和密码登录；
2. 系统成功的验证了用户名密码的匹配；
3. 获取到用户的上下文信息（角色列表等）；
4. 简历这个用户的安全上下文（security context）；
5. 用户可能继续进行一些受到访问控制机制保护的操作（CURD等），访问控制机制会依据当前安全上下文信息检查这个操作所需的权限。

1 - 3是验证过程，过程主要有：

1. 用户和密码被获取到，并放入一个UsernamePasswordAuthenticationToken实例中；
2. 这个token被传递到一个AuthenticationManager实例中进行验证；
3. 在验证成功后，AuthenticationManager返回一个所有字段都被赋值的Authentication对象实例；
4. 通过调用SecurityContextHolder . getContext( ) . setAuthentication( ... )创建全局上下文，通过返回的验证对象进行传递。

从这个角度来说，用户被认为已经验证成功。

事实上，Spring Security 并不关心你如何将Authentication对象放入SecurityContextHolder中。唯一的关键要求是在AbstractSecurityInterceptor验证一个用户请求之前确保SecurityContextHolder包含一个用于表示principal的Authentication对象。

所以我们只要编写自己的Filter或者MVC Controller 从某个位置读取第三方用户信息，构建一个特定的Spring Security Authentication对象，并将其放入SecurityContextHolder中。在这种情况下，需要考虑基础结构上自动应用这些。