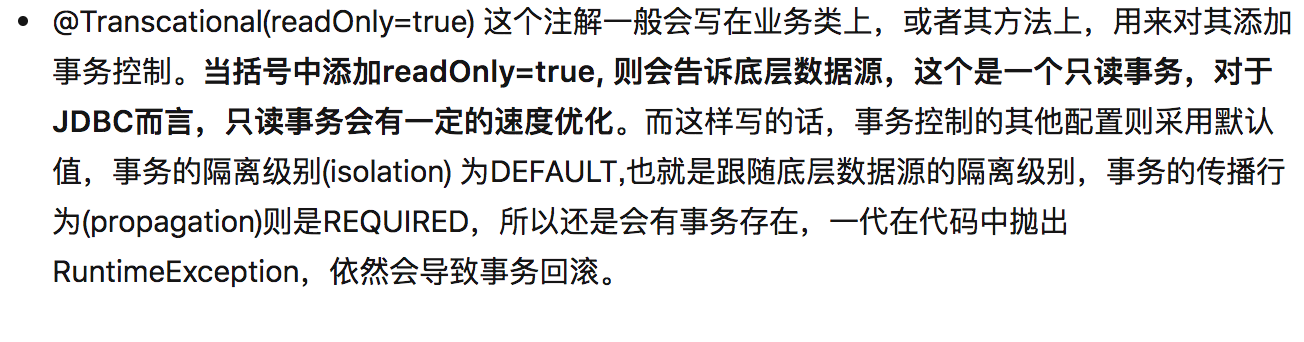
Validation



SecurityConfig



@Transactional(readOnly = true)



Authentication

Authentication是一个接口，用来表示用户认证信息的，在用户登录认证之前相关信息会封装为一个Authentication具体实现类的对象，在登录认证成功之后又会生成一个信息更全面，包含用户权限等信息的Authentication对象，然后把它保存在SecurityContextHolder所持有的SecurityContext中，供后续的程序进行调用，如访问权限的鉴定等。

SecurityContextHolder

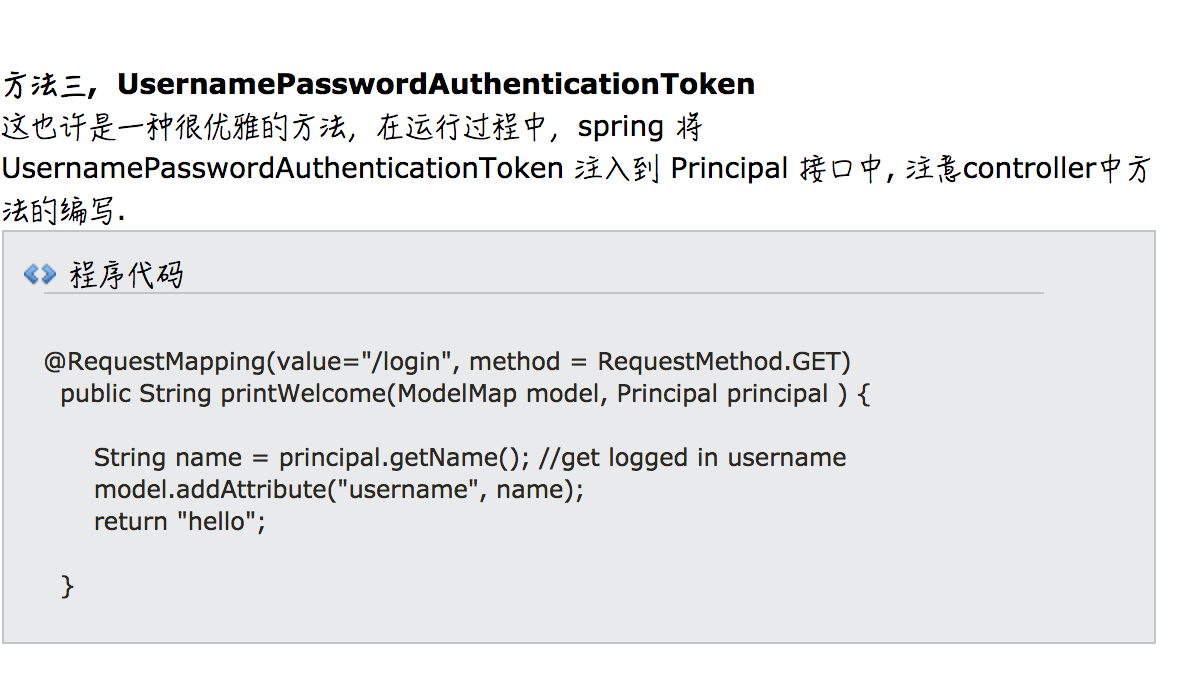
SecurityContextHolder是用来保存SecurityContext的。SecurityContext中含有当前正在访问系统的用户的详细信息。默认情况下，SecurityContextHolder将使用ThreadLocal来保存SecurityContext，这也就意味着在处于同一线程中的方法中我们可以从ThreadLocal中获取到当前的SecurityContext。因为线程池的原因，如果我们每次在请求完成后都将ThreadLocal进行清除的话，那么我们把SecurityContext存放在ThreadLocal中还是比较安全的。这些工作Spring Security已经自动为我们做了，即在每一次request结束后都将清除当前线程的ThreadLocal。

       SecurityContextHolder中定义了一系列的静态方法，而这些静态方法内部逻辑基本上都是通过SecurityContextHolder持有的SecurityContextHolderStrategy来实现的，如getContext()、setContext()、clearContext()等。而默认使用的strategy就是基于ThreadLocal的ThreadLocalSecurityContextHolderStrategy。另外，Spring Security还提供了两种类型的strategy实现，GlobalSecurityContextHolderStrategy和InheritableThreadLocalSecurityContextHolderStrategy，前者表示全局使用同一个SecurityContext，如C/S结构的客户端；后者使用InheritableThreadLocal来存放SecurityContext，即子线程可以使用父线程中存放的变量。

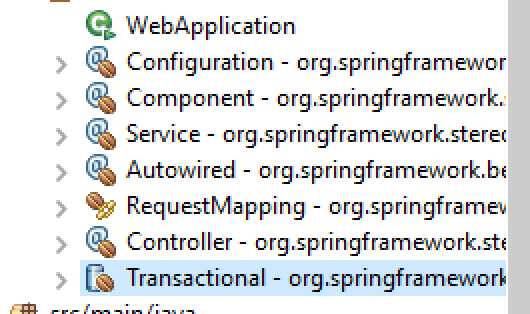
       一般而言，我们使用默认的strategy就可以了，但是如果要改变默认的strategy，Spring Security为我们提供了两种方法，这两种方式都是通过改变strategyName来实现的。SecurityContextHolder中为三种不同类型的strategy分别命名为MODE\_THREADLOCAL、MODE\_INHERITABLETHREADLOCAL和MODE\_GLOBAL。第一种方式是通过SecurityContextHolder的静态方法setStrategyName()来指定需要使用的strategy；第二种方式是通过系统属性进行指定，其中属性名默认为“spring.security.strategy”，属性值为对应strategy的名称。

       Spring Security使用一个Authentication对象来描述当前用户的相关信息。SecurityContextHolder中持有的是当前用户的SecurityContext，而SecurityContext持有的是代表当前用户相关信息的Authentication的引用。这个Authentication对象不需要我们自己去创建，在与系统交互的过程中，Spring Security会自动为我们创建相应的Authentication对象，然后赋值给当前的SecurityContext。但是往往我们需要在程序中获取当前用户的相关信息，比如最常见的是获取当前登录用户的用户名。在程序的任何地方，通过如下方式我们可以获取到当前用户的用户名。

UsernamePasswordAuthenticationToken

UsernamePasswordAuthenticationToken 注入到 Principal 接口中, 注意controller中方法的编写.

Bug



需要 Transactional-。。。。。。

And Spring Data。。。。。

Delete 2 lines

