轻松入门SpringSecurity

讲师：李新杰

问题(Question)

Admin角色应该能访问所有的url

url和角色（权限）的对应关系应该动态加载

本质(Essence)

场景：
某个用户要访问某个url

问题：
make a decision，whether or not allowed

实现：
获取允许访问这个url的一或多个角色
获取这个用户具有的一或多个角色
检查用户是否至少具有一个符合的角色

权限(Auth)

ConfigAttribute：表示访问一个url需要的权限（角色）

GrantedAuthority：表示一个用户被授予的权限（角色）

一个url通常对应一个权限集合，即Collection<ConfigAttribute>

一个用户通常拥有一个权限集合，即Collection<GrantedAuthority>

看这两个集合是否有交集

权限集合的获取(How to get)

Collection<ConfigAttribute> ，根据事先配置好的url和权限的对应关系，在一个请求到来时由系统提供

Collection<GrantedAuthority> ，在用户登陆时，根据用户名从数据库中查询出来，其实是由我们自己提供

做出决定(How to make a decision)

人类社会中，在做出重大决定时可以让多个投票者进行投票

投票者(Voter)

AccessDecisionVoter：投票者，可以投三种票，通过/拒绝/弃权

做出决定(How to make a decision)

投票者众多，容易混乱，所以需要一个管理者来整合投票结果，做出最终裁决

管理者(Manager)

AccessDecisionManager：管理多个投票者，根据投票结果做出决定

管理员角色(ROLE_Admin)

要求：只要拥有管理员角色，直接放行

实现：定义一个基于角色的投票者，检查当前用户是否具有管理员角色。有则投通过，允许访问；无则投弃权，让后续投票者继续投票

基于URI的验证(Uri-Based Check)

情景：以前的小系统，用户登陆时，会把用户可以访问的uri查出来保存在session里。当用户执行操作时，获取当前url，并与用户可访问的uri进行比对，来决定用户是否可以执行此操作

实现：定义一个基于URI的投票者，从request中获取当前请求url，并去用户的权限集合中进行比对，有则投通过，允许访问，无则投拒绝，无权访问

管理员(Manager)

要求：管理多个投票者，做出决定

实现：定义一个管理员，它里面包含多个投票者，一次取出一个投票者进行投票，如果投的是通过，则做出允许访问的决定；如果投的是拒绝，则做出拒绝访问的决定；如果投的是弃权，则取出下一个投票者进行投票。

根据业务需要，决定投票策略，可以一票通过；一票否决；以多胜少；超过指定百分比等。

注意(Attention)

框架已经提供了默认的投票者和管理员，现在我们使用了自定义的投票者和管理员，就等于把默认的做出决定这部分的逻辑给替换了，为了保证系统正常运行，我们还需提供一些额外的支持。

如支持permitAll， authenticated ，hasRole(‘ROLE_User’)，hasAnyRole(‘ROLE_A’, ‘ROLE_B’)等。

对内建权限的支持(Support)

要求：支持系统的内建权限，如permitAll，authenticated等。

实现：定义一个投票者，当检测到这些内建权限时，按要求进行投票即可。

普通用户角色(ROLE_User)

要求：支持系统里默认的角色表达式，如hasRole(‘ROLE_User’)，hasAnyRole(‘ROLE_A’, ‘ROLE_B’)

实现：在前面那个基于角色的投票者里，获取角色表达式，从中解析出角色字符串，去用户具有的角色集合里比对。有则投通过，允许访问；无则投弃权，让后续投票者继续投票

问题(Question)

url和角色（权限）的对应关系应该动态加载