其实不管基于内存的操作还是基于数据库的操作，security的目的都是只有一个，拿到当前的User以及相关的UserDetails信息，在基于内存的时候，用户名以及相关的UserDetails都会存到内存中，同理，如果迁移到数据库中，那么，我们可以从数据库中查出当前User以及相关的用户信息，然后封装成一个实现了UserDetails接口的User实现类。理解到这里，目标已经非常明确了：返回一个实现了UserDetails的User类给框架。

在security框架中，无论是InMemoryDaoImpl还是基于数据库的实现类都是利用同一个接口UserDetailService的方法loadUserByUsername。（这种面向接口的设计方式是非常棒的，纯属个人的感悟）。这里我们只需要改变这个方法的实现方式就可以达到目的了。

**1.5**     **GrantedAuthority**

       Authentication的getAuthorities()可以返回当前Authentication对象拥有的权限，即当前用户拥有的权限。其返回值是一个GrantedAuthority类型的数组，每一个GrantedAuthority对象代表赋予给当前用户的一种权限。GrantedAuthority是一个接口，其通常是通过UserDetailsService进行加载，然后赋予给UserDetails的。

       GrantedAuthority中只定义了一个getAuthority()方法，该方法返回一个字符串，表示对应权限的字符串表示，如果对应权限不能用字符串表示，则应当返回null。

       Spring Security针对GrantedAuthority有一个简单实现SimpleGrantedAuthority。该类只是简单的接收一个表示权限的字符串。Spring Security内部的所有AuthenticationProvider都是使用SimpleGrantedAuthority来封装Authentication对象。

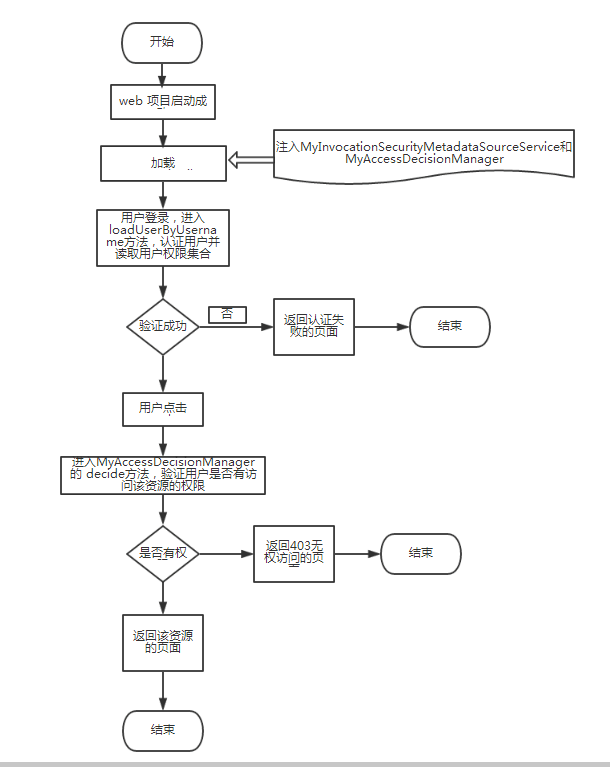
下面总结下authenticationProvider：   
1、验证过程中，可以增加多个authenticationProvider，来完成不同的验证工作   
2、根据验证的先后顺序，需要注意authenticationProvider设置的先后顺序。   
3、如果抛出异常，即可停止的话，不能抛出AuthenticationException   
4、每个authenticationProvider都要配置一个UserDetailsService的实现类   
5、如果不使用DaoAuthenticationProvider或要重新配置，就直接使用auth.authenticationProvider()方法，不要使用auth .userDetailsService()

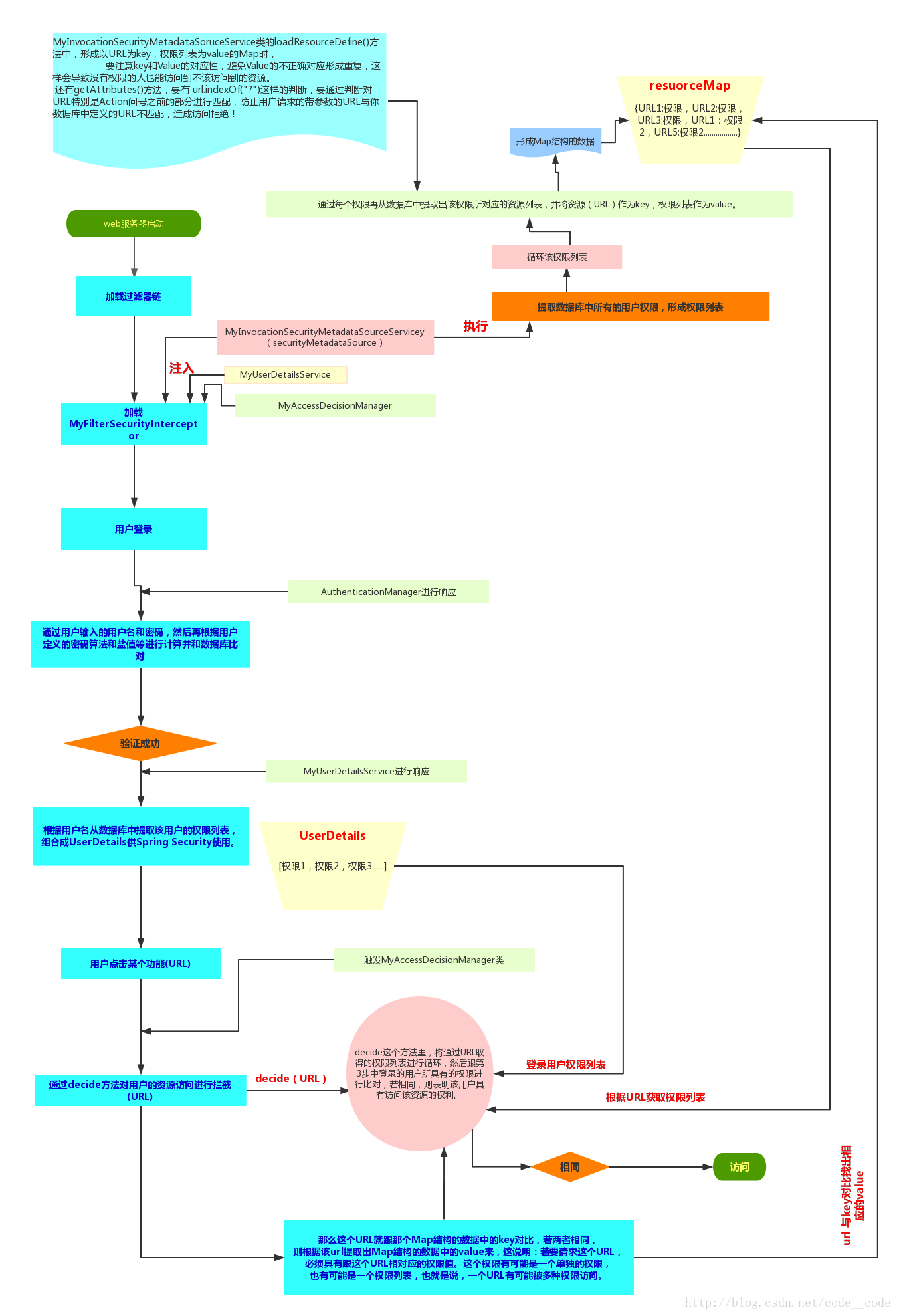
很重要的依赖巨坑：我踩！！！

<parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <!--<version>2.0.1.RELEASE</version>-->  
 <version>1.4.3.RELEASE</version> <!-- !!!!!!!!该版本号，很重要！！！！-->  
 <relativePath/> <!-- lookup parent from repository -->  
</parent>

同时：

<!--!!!!!!!!!!!!!!该依赖不能加，加了会导致jpa映射出错，导致外键约束失效。  
要保证jpa映射，外键约束正常，还要同时与上面spring-boot-starter-parent 版本号要一致！！！！！！！！！！！！ -->  
<!--<dependency>-->  
 <!--<groupId>org.springframework.security</groupId>-->  
 <!--<artifactId>spring-security-test</artifactId>-->  
 <!--<scope>test</scope>-->  
<!--</dependency>-->





jpa外键关联：

JPA总结——实体关系映射（一对多@OneToMany） - CSDN博客

<https://blog.csdn.net/caiyanzhi123/article/details/50828187>

FetchType.LAZY和FetchType.EAGER什么区别？（懒加载和急加载的理解） - CSDN博客

<https://blog.csdn.net/u010082453/article/details/43339031>