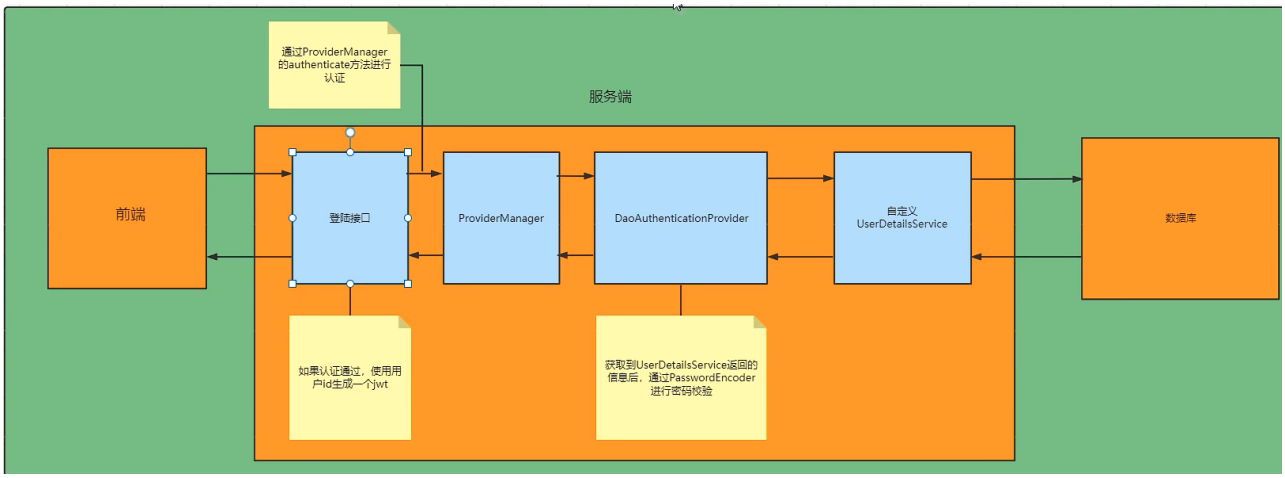
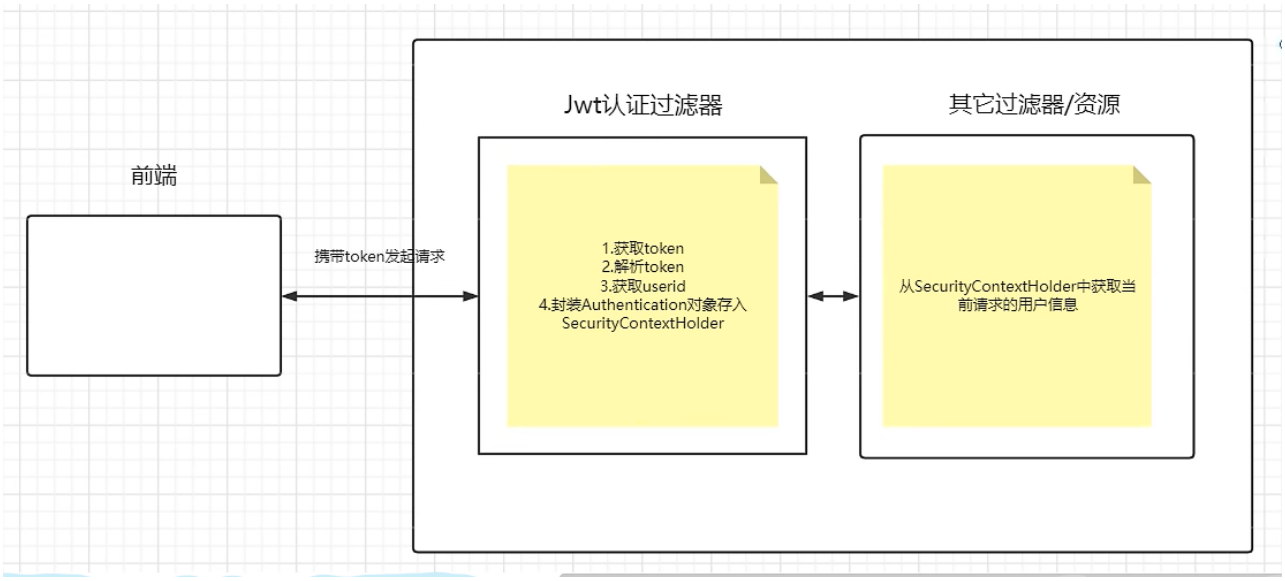
1. 认证：验证当前访问系统的是不是本系统的用户，并且要确认具体是哪个用户
2. 授权：经过认证后判断当前用户是否有权限进行某个操作
3. 创建项目，添加依赖，引入security
4. 前后端登录流程
    a. 携带用户名和密码访问登录接口
    b. 数据库比对进行校验
    c. 正确后，使用用户名登资料，生成jwt
    d. 把jwt响应给前端
    e. 登录后访问其他请求需要再请求头中携带token
    f. 获取请求头中的token进行解析，获取user
    g. 根据user获取用户信息，如果有权限则允许访问相关资源
    h. 访问目标资源，响应给前端
    i. 返回响应信息
    j.





5. 完整流程（过滤器链），以下是核心过滤器，14个过滤器
    a. UsernamePasswordAuthenticationFilter：负责处理我们在登录页填写用户名密码后的登录请求

i. 封装Authentication对象，只有用户民和密码，没有权限

ii. 调用ProviderManager的authenicate方法进行认证

iii. 调用DaoAuthenticationProvider的authenticate的方法进行认证

iv. 调用loadUserByUsername方法查询用户

v. 根据用户名查询对应的用户以及这个用户的对应的权限信息，InMemoryUserDetailsManager是在内存中查找

vi. 把对应的用户信息包括他的权限信息封装成UserDetails对象

vii. 返回UserDetails对象

viii. 通过PasswordEncoder对比UserDetails中的密码和Authentication的密码是否正确

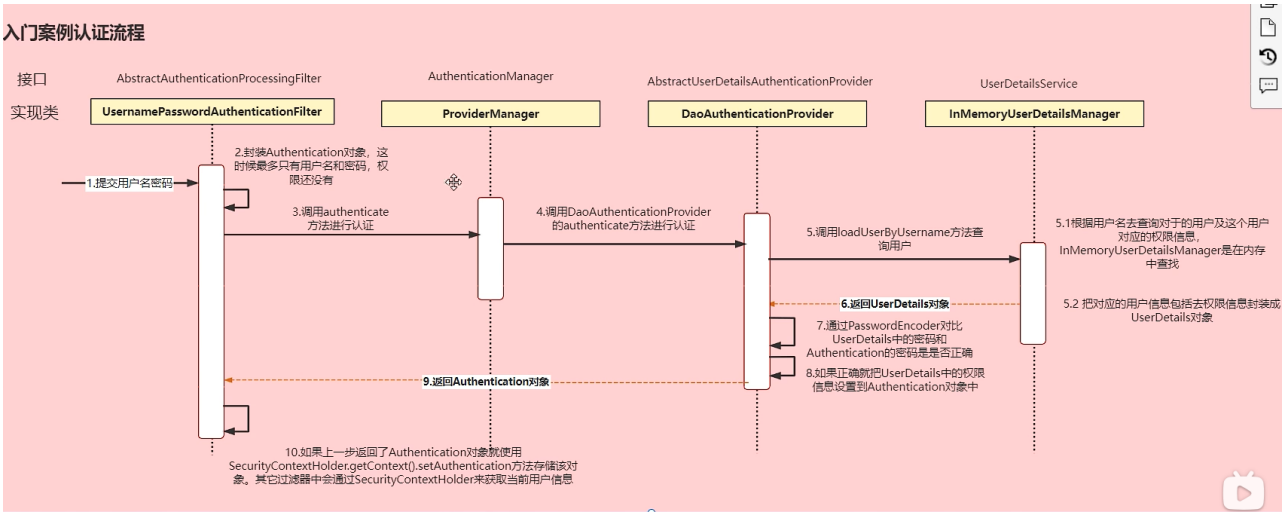
ix. 如果正确就把UserDetails中的权限信息设置到Authentication对象中

x. 返回Authentication对象

xi. 如果返回Authentication对象，就使用SecurityContextHolder的getContext的setAuthentication方法存储该对象，

xii. 其他过滤器会通过SecurityContextHolder来获取当前用户信息

xiii.



b. ExceptionTranslationFilter：处理过滤器链中抛出的任何AccessDeniedException和AuthenticationException

c. FilterSecutrityInterceptor：负责权限校验的过滤器

6. JWT

a. 全称是JSON Web Token，通过JSON形式作为web应用中的令牌，用于在各方面之间安全的将信息作为json对象传输，在数据传输过程中还可以完成数据加密，签名等处理

b. 可以通过URL，POST参数或者在HTTP hender发送

c. 负载（payload）包含用户所需信息，避免多次查询数据库

d. 以JSON加密形式保存在客户端，跨语言

e. 不需要在服务端保存会话信息，使用分布式微服务

f. 由标头（Header），有效载荷（Payload），签名（Signature）组成

i. 标头通常由两部分组成，令牌类型和所使用的签名算法，使用Base64编码组成JWT结构一部分

ii. 有效载荷包含声明，是有关实体和其他数据的声明，使用Base64编码组成JWT结构第二部分

iii. 编码后的标头和有效载荷以及提供的密钥（随机盐值），使用标头指定的签名算法进行签名，保证JWT没有被篡改过

7. 重写UserDetailsService的load 方法
8. 创建自己的user实体类，实现UserDetails
9.