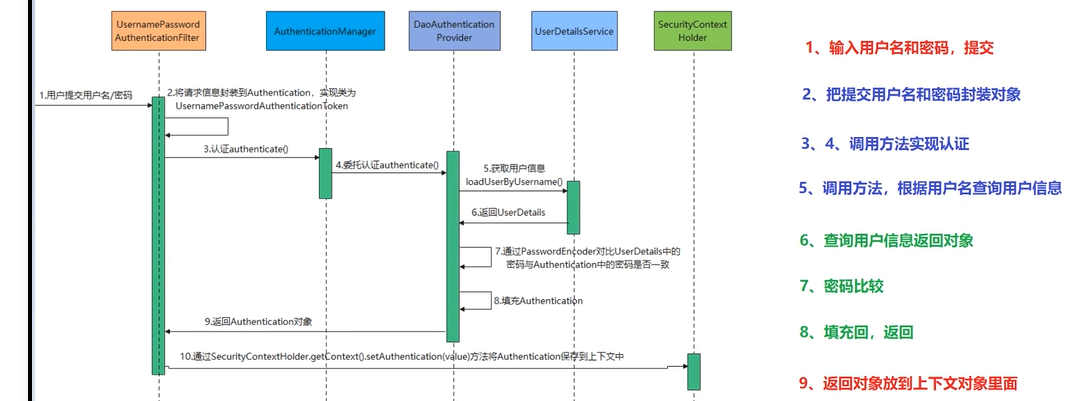
Spring-security



The green part is responsible for authentication filters,

The blue part is responsible for exception handling

The orange part is responsible for authorization



Three core components in Spring Security:

​ 1. Authentication：The authentication information is stored, representing the current logged-in user

Information included：

(1). Principal: User information, such as the user name without authentication. After authentication, the user object is stored.

(2). Credentials: User credentials, usually a password.

(3). Authorities: Control authority

​ 2. SeucirtyContext：Context object, used to get Authentication

​ 3. SecurityContextHolder: Context management object, used to get “SecurityContext” anywhere in the program

How does Spring Security authenticate users?

'AuthenticationManager' is the component that Spring Security uses to perform authentication, and simply calls its' authenticate 'method to complete the authentication. Spring Security authentication is by default in ` UsernamePasswordAuthenticationFilter ` authentication in the filter, the filter is responsible for the authentication logic.

1. Step 7,8,9,10 finished by spring-security so the rest is to design a custom component based on our User entity, MD5 password encryption, and loadUserByUsername() method.

The logic for AuthenticationManager: The user object is queried based on the user name. If no user object is found, an exception is thrown. The password of the user object is verified with the passed password. If the password does not match, an exception is thrown

Custom component:

1. UserDetails : In actual development, we have a variety of user properties, and these default properties may not be satisfied, so we generally implement the interface ourselves, and then set up our actual user entity object.
2. LoadByUserName: Handled by “UserDetialsService”, the interface has only one method' loadUserByUsername(String username) ', which queries the user object by the username.
3. PasswordEncoder: Responsible for password encryption and verification

Filter:

TokenLoginFilter: Extends UsernamePasswordAuthenticationFilter: See Step 1 in the figure above.

TokenAuthenticationFilter: Because the user login status is stored in the token on the client side, the request header of each request interface carries the token. The background uses a customized token filter to intercept and parse the token to complete authentication and fill the user information entity(因为用户登录状态在token中存储在客户端，所以每次请求接口请求头携带token， 后台通过自定义token过滤器拦截解析token完成认证并填充用户信息实体)

Config: Fixed format. In my test environment, only the login interface does not require a token