1.1 用户名密码登录逻辑

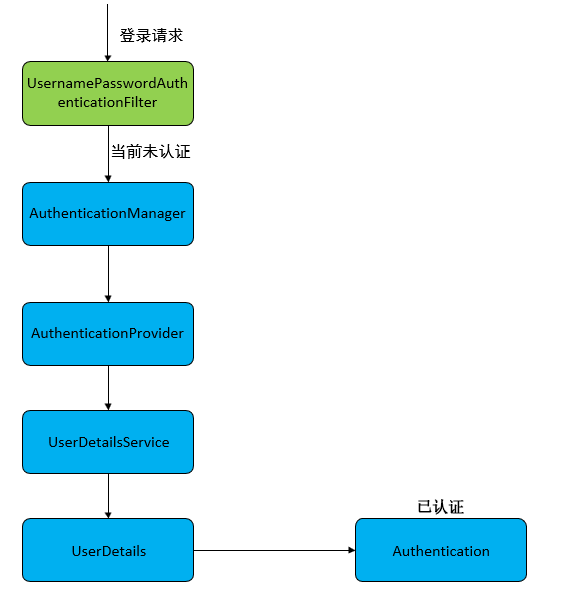
以标准的用户名密码登录为例，讲解了整个认证流程。大致流程如下：

先进入 UsernamePasswordAuthenticationFilter 中，根据输入的用户名和密码信息，构造出一个暂时没有鉴权的 UsernamePasswordAuthenticationToken，并将 UsernamePasswordAuthenticationToken 交给 AuthenticationManager 处理。

AuthenticationManager 本身并不做验证处理，他通过 for-each 遍历找到符合当前登录方式的一个 AuthenticationProvider，并交给它进行验证处理，对于用户名密码登录方式，这个 Provider 就是 DaoAuthenticationProvider。

在这个 Provider 中进行一系列的验证处理，如果验证通过，就会重新构造一个添加了鉴权的 UsernamePasswordAuthenticationToken，并将这个 token 传回到 UsernamePasswordAuthenticationFilter 中。

在该 Filter 的父类 AbstractAuthenticationProcessingFilter 中，会根据上一步验证的结果，跳转到 successHandler 或者是 failureHandler。



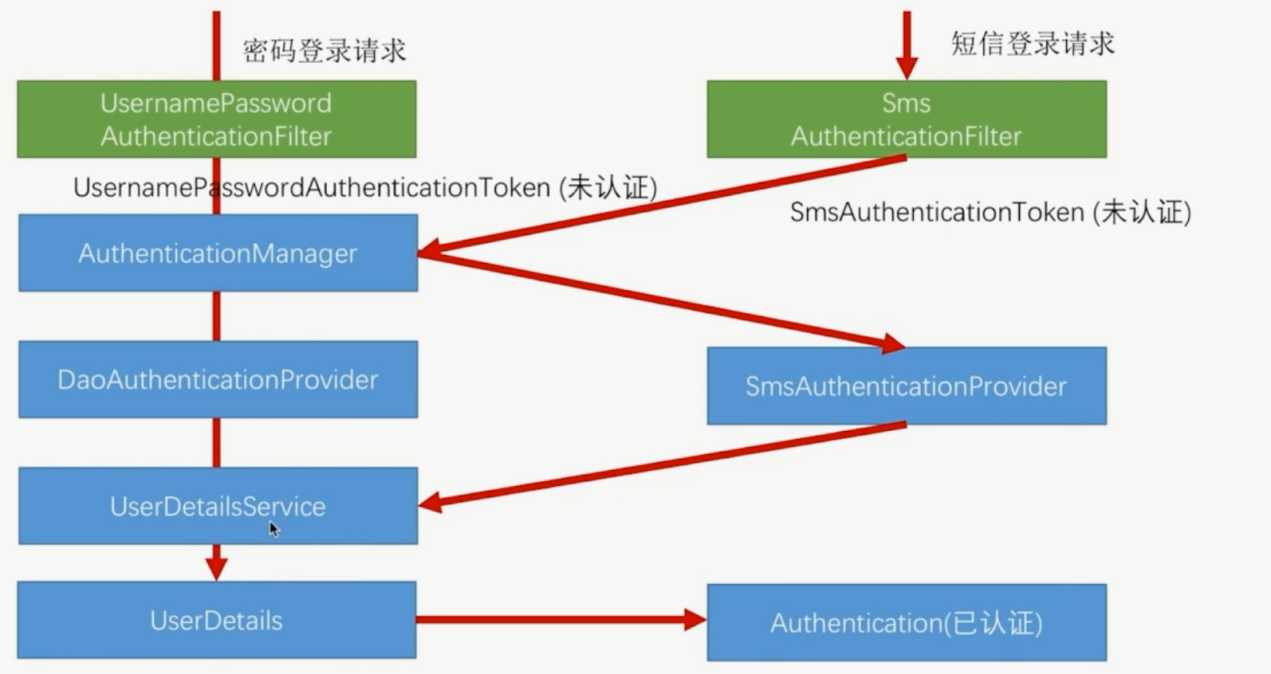
1.2 短信验证码登录逻辑

我们可以仿照用户名密码登录的逻辑，来实现短信验证码的登陆逻辑。

用户名密码登录有个 UsernamePasswordAuthenticationFilter ，我们搞一个 SmsAuthenticationFilter，代码粘过来改一改。

用户名密码登录需要 UsernamePasswordAuthenticationToken，我们搞一个 SmsAuthenticationToken，代码粘过来改一改。

用户名密码登录需要 DaoAuthenticationProvider，我们模仿它也 implenments AuthenticationProvider，叫做 SmsAuthenticationProvider。



我们自己搞了上面三个类以后，想要实现的效果如上图所示。当我们使用短信验证码登录的时候：

先经过 SmsAuthenticationFilter，构造一个没有鉴权的 SmsAuthenticationToken，然后交给 AuthenticationManager 处理。

AuthenticationManager 通过 for-each 挑选出一个合适的 provider 进行处理，当然我们希望这个 provider 要是 SmsAuthenticationProvider。

验证通过后，重新构造一个有鉴权的 SmsAuthenticationToken，并返回给 SmsAuthenticationFilter。

filter 根据上一步的验证结果，跳转到成功或者失败的处理逻辑。