（2）如果 authRequest 变量为 null，说明请求头中没有包含认证信息，那么直接执行接下来的过滤器即可，该方法也到此为止。在执行接下来的过滤器时，最终就会通过 ExceptionTranslationFilter 过滤器进入到 BasicAuthenticationEntryPoint#commence 方法中；如果 authRequest 变量不为 null，说明请求是携带了认证信息的，那么就对请求携带的认证信息进行校验。

（3）从 authRequest 对象中提取出用户名，然后调用 authenticationIsRequired 方法判断是否有必要进行认证，如果没有必要，则直接执行剩下的过滤器即可；如果有必要进行认证，则进行用户认证。authenticationIsRequired 方法的具体逻辑就是，从 SecurityContextHolder 中取出当前登录对象，判断使用是否已经登录过了，同时判断是否就是当前用户。

（4）如果有必要进行认证，则调用 authenticationManager.authenticate 方法完成用户认证，同时将用户信息存入 SecurityContextHolder；如果配置了 rememberMeServices，也进行相应的处理，最后还有一个登录成功的回调方法 onSuccessfulAuthentication，不过该方法并未做任何实现。

（5）如果认证过程抛出异常，则进行相应处理即可，这里逻辑比较简单。

（6）最后继续执行接下来的过滤器。在后续过滤器的执行过程中，由于 SecurityContextHolder 中已经保存了登录用户信息了，相当于用户已经完成登录了，因此就和普通的请求一致，不会被“半路拦截”。

这就是整个 HTTP 基本认证的实现逻辑，可以看到实现还是比较容易的。

10.2 HTTP Digest authentication

10.2.1 简介

HTTP 基本认证虽然简单易用，但是在安全方面问题突出，于是又推出了 HTTP 摘要认证。HTTP 摘要认证最早在 RFC2069 中被定义，随后被 RFC2617（https://tools.ietf.org/html/rfc2617）所取代，在 RFC2617 中引入了一系列增强安全性的参数，以防止各种可能存在的网络攻击。

相比于 HTTP 基本认证，HTTP 摘要认证的安全性有了很大提高，但是依然存在问题，例如不支持 bCrypt、PBKDF2、SCrypt 等加密方式。

图 10-3 所示描述了 HTTP 摘要认证的具体流程，从图中可以看出，这个认证流程和 HTTP 基本认证流程一致，不同的是每次传递的参数有所差异。