2-12 刷新令牌应该如何设计？

一、为什么要刷新Token的过期时间？

我们在定义JwtUtil工具类的时候，生成的 `Token` 都有过期时间。那么问题来了，假设 `Token` 过期时间为15天，用户在第14天的时候，还可以免登录正常访问系统。但是到了第15天，用户的Token过期，于是用户需要重新登录系统。

`HttpSession` 的过期时间比较优雅，默认为15分钟。如果用户连续使用系统，只要间隔时间不超过15分钟，系统就不会销毁 `HttpSession` 对象。JWT的令牌过期时间能不能做成 `HttpSession` 那样超时时间，只要用户间隔操作时间不超过15天，系统就不需要用户重新登录系统。实现这种效果的方案有两种： `双Token` 和 `Token缓存` ，这里重点讲一下 `Token` 缓存方案。

Token缓存方案是把 `Token` 缓存到Redis，然后设置Redis里面缓存的 `Token` 过期时间为正常 `Token` 的1倍,然后根据情况刷新 `Token` 的过期时间。

Token失效，缓存也不存在的情况

当第15天，用户的 `Token` 失效以后，我们让Shiro程序到Redis查看是否存在缓存的 `Token` ，如果这个 `Token` 不存在于Redis里面，就说明用户的操作间隔了15天，需要重新登录。

Token失效，但是缓存还存在的情况

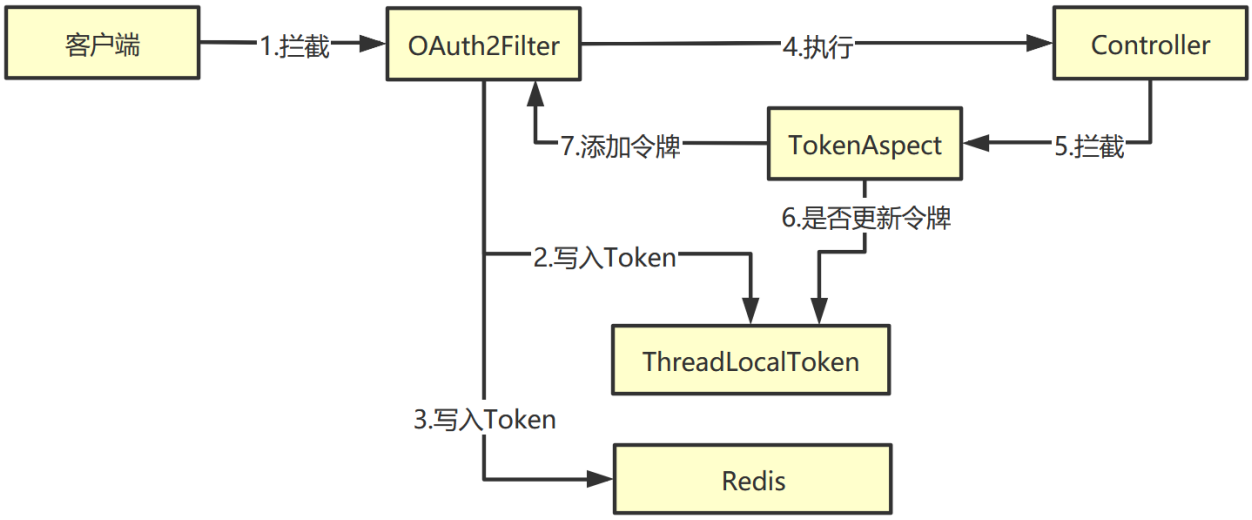
如果Redis中存在缓存的 `Token` ，说明当前 `Token` 失效后，间隔时间还没有超过15天，不应该让用户重新登录。所以要生成新的 `Token` 返回给客户端，并且把这个 `Token` 缓存到Redis里面，这种操作成为刷新 `Token` 过期时间。

二、客户端如何更新令牌？

在我们的方案中，服务端刷新 `Token` 过期时间，其实就是生成一个新的 `Token` 给客户端。那么客户端怎么知道这次响应带回来的Token是更新过的呢？这个问题很容易解决。

只要用户成功登陆系统，当后端服务器更新 `Token` 的时候，就在响应中添加 `Token` 。客户端那边判断每次Ajax响应里面是否包含 `Token` ，如果包含，就把 `Token` 保存起来就可以了。

三、如何在响应中添加令牌？



我们定义 `OAuth2Filter` 类拦截所有的HTTP请求，一方面它会把请求中的 `Token` 字符串提取出来，封装成对象交给Shiro框架；另一方面，它会检查 `Token` 的有效性。如果 `Token` 过期，那么会生成新的 `Token` ，分别存储在 `ThreadLocalToken` 和 `Redis` 中。

之所以要把 `新令牌` 保存到 `ThreadLocalToken` 里面，是因为要向 `AOP切面类` 传递这个 `新令牌` 。虽然 `OAuth2Filter` 中有 `doFilterInternal()` 方法，我们可以得到响应并且写入 `新令牌` 。但是这个做非常麻烦，首先我们要通过IO流读取响应中的数据，然后还要把数据解析成JSON对象，最后再放入这个新令牌。如果我们定义了 `AOP切面类` ，拦截所有Web方法返回的 `R对象` ，然后在 `R对象` 里面添加 `新令牌` ，这多简单啊。但是 `OAuth2Filter` 和 `AOP` 切面类之间没有调用关系，所以我们很难把 `新令牌` 传给 `AOP切面类` 。

这里我想到了 `ThreadLocal` ，只要是同一个线程，往 `ThreadLocal` 里面写入数据和读取数据是完全相同的。在Web项目中，从 `OAuth2Filter` 到 `AOP切面类` ，都是由同一个线程来执行的，中途不会更换线程。所以我们可以放心的把新令牌保存都在 `ThreadLocal` 里面， `AOP切面类` 可以成功的取出新令牌，然后往 `R对象` 里面添加新令牌即可。

`ThreadLocalToken` 是我自定义的类，里面包含了 `ThreadLocal` 类型的变量，可以用来保存线程安全的数据，而且避免了使用线程锁。