链滴

spring security 之通过 token 验证用户保持登录状态

作者：524021835

原文链接：https://ld246.com/article/1545318463746

来源网站：链滴

第一次使用springSecurity，采用前后端分离的方式，vue多web端通过正常方式访问，App端需要设计为不强制登录（只要不是App安装、版本更新、注销登录，在第一次登录后不需要重次登录），由于security默认的登录方式不支持这方式，需要重写过滤器，修改为支持从header中获取token，根据token设置当前登录对象。

思路大概为：App端登录成功，后端返回token值，App保存在本地后在后面的每次请求中都在header中带着token进行求，后端重写过滤器，在过滤器中解析token值并将token带的对象放到登录用户中，让security认为请求已经是在登状态下进行。

1、新建AuthenticationTokenFilter继承OncePerRequestFilter，重写doFilterInternal方法。在doFilterInternal中获取header中带着的验证信息，解析token值获取用户信息，并将用户信息设置到SecurityContextHolder中。

```
@Component
public class AuthenticationTokenFilter extends OncePerRequestFilter {


    @Autowired
  private UserService userService; //用户信息service

  @Override
  protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, F
lterChain chain) throws ServletException, IOException {
        String authHeader = request.getHeader("token");//获取header中的验证信息

  if (authHeader != null && authHeader.startsWith("Bearer ")) {
          final String authToken = authHeader.substring("Bearer ".length());

  String username = JwtUtils.parseToken(authToken, "\_secret"); //从token中获取用户信息，jw
Utils自定义的token加解密方式

  if (username != null && SecurityContextHolder.getContext().getAuthentication() == null) {
              UserDetails userDetails = userService.loadUserByUsername(username);//根据用户名
取用户对象

  if (userDetails != null) {
                  UsernamePasswordAuthenticationToken authentication =
                        new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.g
tAuthorities());
  authentication.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));
  //设置为已登录
  SecurityContextHolder.getContext().setAuthentication(authentication);
  }
          }
        }

        chain.doFilter(request, response);
  }
}
```

2、在继承了WebSecurityConfigurerAdapter的类中增加以下内容

```
@Autowired
AuthenticationTokenFilter authenticationTokenFilter; // token 拦截器

@Override
protected void configure(HttpSecurity http) throws Exception {
// 加上addFilterBefore执行token拦截器
    http
            .addFilterBefore(authenticationTokenFilter, UsernamePasswordAuthenticationFilter.cla
s)
...
```

参考内容：https://blog.csdn.net/larger5/article/details/81063438