基于Spring Security和JWT实现无状态登录

王萍
（南京工业职业技术大学 计算机与软件学院，江苏 210023）

摘要：阐述用户登录Web多次请求通过无状态登录，使客户端请求不依赖服务端的信息，可以更方便地进行集群化部署，减小服务端存储压力，从而实现了真正意义上的前后端分离。

关键词：计算机工程，用户登录，无状态，集群化部署。
中图分类号：TP309，TP393.08

文章编号：1000-0755(2021)12-0114-02
文献引用格式：王萍.基于Spring Security和JWT实现无状态登录[J].电子技术, 2021, 50(12): 114-115.

Stateless Login Based on Spring Security and JWT

WANG Ping
(School of Computer and Software, Nanjing Polytechnic University, Jiangsu 210023, China.)

Abstract — This paper expounds that users log in to the web for multiple requests. Through stateless login, the client request does not depend on the information of the server, which can be more convenient for cluster deployment and reduce the storage pressure of the server, so as to realize the real separation of the front and back ends.

Index Terms — computer engineering, user login, stateless, cluster deployment.

0 引言

随着技术的不断进步，web项目的开发从传统基于Servlet技术的开发模式转变为前后端分离的开发模式。在前后端分离模式下，前端人员只需要关注数据的解析以及页面的渲染，后端人员只需要提供符合约定的数据，从技术角度来讲实现前端技术和后端技术的解耦，从业务逻辑上来讲，是业务逻辑和用户体验的解耦。前后端以并行开发来提高开发效率。其中REST架构是目前前后端分离的最佳实践，目前已成为构建web服务时应该遵循的事实标准。

1 “状态”的概念

Web应用程序协议的“状态”通常指的是相互关联的用户交互操作保留的一些公共信息，它们通常被用来存储工作流或用户状态信息等相关数据。这些信息在web开发中可以被指定为不同的作用域如page、request、session、application，它们的存储由客户端或服务器端负责。

服务调用过程中有两种状态，分别是应用状态和资源状态。应用状态是指与某一特定请求有关的状态信息，而资源状态是存储在服务器端资源在某一时刻的特定状态，它不会因为用户请求而发生更改，任一用户在同一时刻对该资源的请求都会获得这一状态的表现。在有状态的模式下，一个用户的请求必须被提交到保存有相关信息的服务器上，否则这些请求可能无法被理解，也就意味着在此模式下服务器端无法对用户的请求进行自由调度。

2 “无状态”登录

无状态指的是任意一个web请求必须完全与其他请求相隔离，当请求端提出请求时，该请求本身包含了相应端为相应请求所需要的全部信息。REST架构要求服务器端不保存任何与特定HTTP请求相关的资源，应用状态必须由请求方在请求过程中提供。其中，用户的身份凭证信息作为一种应用状态，是被期望由请求方提供的，所以在请求中传递用户的身份凭证信息是符合REST架构规范的。

3 用户登录功能的实现

基于SpringBoot采用Spring Security+JWT安全框架实现用户登录功能，其中SpringSecurity实现拦截用户请求、校验用户令牌和提供授权，而JWT负责定义令牌格式、确定令牌校验规则和保证令牌安全性。其解决方案如图1所示。

后端关键代码如下：

```
    public String createToken(Authentication authentication) {
    String authorities = authentication.getAuthorities().stream().map(GrantedAuthority::getAuthority).collect(Collectors.joining(","));
    long now = (new Date()).getTime();
            Date validity = new Date(now + this.tokenValidityInMillisecondsForRememberMe);
    return Jwts
    .builder()
    .setSubject(authentication.getName())
    .claim(AUTHORITIES_KEY, authorities)
    .signWith(key, SignatureAlgorithm.HS512)
    .setExpiration(validity)
    .compact();
    }
    public Authentication getUser(String token) {
```

作者简介：王萍，南京工业职业技术大学计算机与软件学院，讲师，硕士；研究方向：软件技术。
收稿日期：2021-09-26；修回日期：2021-12-11。

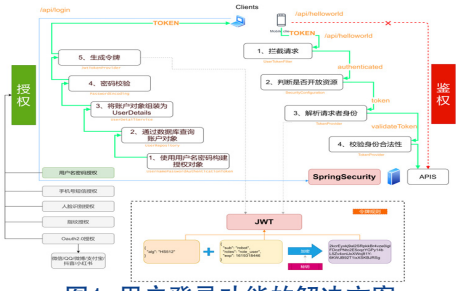

图1 用户登录功能的解决方案

```
    Claims claims = jwtParser.parseClaimsJws(token).
getBody();
    Collection<? extends GrantedAuthority>
authorities = Arrays
    .stream(claims.get(AUTHORITIES_KEY).toString().
split(","))
    .filter(auth -> !auth.trim().isEmpty())
    .map(SimpleGrantedAuthority::new)
    .collect(Collectors.toList());
    User principal = new User(claims.getSubject(),
"", authorities);
    return new UsernamePasswordAuthenticationToken(p
rincipal, token, authorities);
    }
    public boolean validateToken(String authToken) {
    try {
    jwtParser.parseClaimsJws(authToken);
    return true;
    } catch (JwtException | IllegalArgumentException
e) {
    e.printStackTrace();
    }
    return false;
    }

    @Override
    public void doFilter(ServletRequest
servletRequest, ServletResponse servletResponse,
FilterChain filterChain) throws IOException,
ServletException {
    HttpServletRequest httpServletRequest =
(HttpServletRequest) servletRequest;
    // 1、从请求头部找到令牌
    String token = httpServletRequest.
getHeader("Authorization");
    if (token == null) {
    // 如果请求头部没有，从查询参数中找到令牌
    token = httpServletRequest.
getParameter("Authorization");
    }
    if (token != null) {
    // 2、使用TokenProvider校验这个token合法性
    if (tokenProvider.validateToken(token)) {
    // 3、token校验通过，通过token工具类将token转为
Authentication
    Authentication user = tokenProvider.
getUser(token);
    // 4、将用户交给SpringSecurity管理
    SecurityContextHolder.getContext().
setAuthentication(user);
    }
            }
    filterChain.doFilter(servletRequest,
servletResponse);
    }
    }

    @Configuration
    public class SecurityConfiguration extends
WebSecurityConfigurerAdapter {
    @Resource
    private UserTokenFilter userTokenFilter;

    @Bean
    public PasswordEncoder passwordEncoder() {
    // 指定密码加密模式：加盐哈希算法
    return new BCryptPasswordEncoder();
    }
    // 将登录模式设置为无状态模式，关闭跨站脚本攻击功
能
    @Override
    protected void configure(HttpSecurity http)
throws Exception {
    http.sessionManagement()
    .sessionCreationPolicy(SessionCreationPolicy.
STATELESS) // 设置无状态会话
    .and().csrf().disable() // 关闭跨域脚本攻击
    .authorizeRequests() // 开放登录接口
    .antMatchers("/", "/login").permitAll()
    .antMatchers("/admin/**").authenticated()
    .and()
    //将自定义过滤器添加到登录验证之前
    .addFilterBefore(userTokenFilter, UsernamePass
wordAuthenticationFilter.class);;
    }
    }
```

前端采用Vue.js实现，Vue.js是用于构建交互式的Web界面的库，它提供MVVM数据绑定和一个可组合的组件系统，具有简单、灵活的API。从技术上讲，Vue.js集中在MVVM模式上的视图模型层(viewModel)，并通过双向数据绑定连接视图（view）和模型（model）。使用VueCLI创建Vue项目，并结合Node.js实现了前端的部署，其项目的入口程序为main.js；登录页面login.vue的关键代码如下：

```
    //逻辑处理
    methods: {
    //登录方法
    doLogin(){
    if(this.username.length==0 || this.password.
length==0){
    //弹出提示
    uni.showToast({
    title:"请输入用户名密码",
    icon:"none"
        });
    return;
        }
    uni.showLoading({
    title:"登录中"
        })
    //发送登录请求
    uni.request({
    url:this.$serverUrl+"/api/login?login="+this.
username+"&password="+this.password,
    method:"POST",
    success:(result)=>{
    console.log("登录请求发送，收到响应：",result);
    uni.hideLoading();
    if(result.statusCode!=200){
    uni.showToast({
    title:"用户名密码错误",
    icon:"none"
        });

        return;
        }else{
        uni.showToast({
        title:"登录成功"
        });
        uni.setStorageSync("token",result.data);
        setTimeout(()=>{
        uni.navigateTo({
        url:'../index/index'
        })
        },1500);
        }
        }
        });
        }
        }
```

4 结语

用户登录功能是每个web项目必备的，通过无状态登录使客户端请求不依赖服务端的信息，多次请求不需要必须访问到同一台服务器；服务端的集群和状态对客户端是透明的；服务端可以任意地迁移和伸缩，可以更方便地进行集群化部署;减小服务端存储压力，从而实现真正前后端分离。

参考文献

[1] 柳纲,张毅.服务端无状态技术研究[J].电力信息与通信技术,2017,15(11):49-54.

[2] 姚刚,李群,蔡凤翔.浅谈微服务用户认证与访问控制[J].信息系统工程,2021(04):66-68.