链滴

Spring WebFlux REST API Token 登陆

作者：sofior

原文链接：https://ld246.com/article/1553088395299

来源网站：链滴

pom

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
```

TokenAuthentication

- 方便授权时传递token并放入缓存

```
@Data
@EqualsAndHashCode(callSuper = true)
public class TokenAuthentication extends UsernamePasswordAuthenticationToken {

    private final String token;

    public TokenAuthentication(String token, Object principal, Object credentials) {
        super(principal, credentials);
        this.token = token;
    }

    public TokenAuthentication(String token, Object principal, Object credentials, Collection<?
xtends GrantedAuthority> authorities) {
        super(principal, credentials, authorities);
        this.token = token;
    }

}
```

Config Class

```
@EnableWebFluxSecurity
@EnableReactiveMethodSecurity
public class SecurityConfig {
    //可换为redis存储
    private final Map<String, SecurityContext> tokenCache = new ConcurrentHashMap<>();

    private static final String BEARER = "Bearer ";

    private static final String[] AUTH_WHITELIST = new String[]{"/login", "/actuator/**"};

    @Bean
    ReactiveAuthenticationManager reactiveAuthenticationManager() {
        final ReactiveUserDetailsService detailsService = userDetailsService();
        LinkedList<ReactiveAuthenticationManager> managers = new LinkedList<>();
        managers.add(authentication -> {
            //其他登陆方式(比如手机号验证码登陆)可在此设置不得抛出异常或者Mono.error
            return Mono.empty();
        });
        //必须放最后不然会优先使用用户名密码校验但是用户名密码不对时此AuthenticationManage
会调用Mono.error造成后面的AuthenticationManager不生效
```

原文链接：Spring WebFlux REST API Token 登陆

```
        managers.add(new UserDetailsRepositoryReactiveAuthenticationManager(detailsService)
;
        return new DelegatingReactiveAuthenticationManager(managers);
    }

    @Bean
    ServerSecurityContextRepository serverSecurityContextRepository() {
        return new ServerSecurityContextRepository() {
            @Override
            public Mono<Void> save(ServerWebExchange exchange, SecurityContext context) {
                if (context.getAuthentication() instanceof TokenAuthentication) {
                    TokenAuthentication authentication = (TokenAuthentication) context.getAuthent
cation();
                    tokenCache.put(authentication.getToken(), context);
                }
                return Mono.empty();
            }

            @Override
            public Mono<SecurityContext> load(ServerWebExchange exchange) {
                ServerHttpRequest request = exchange.getRequest();
                String authorization = request.getHeaders().getFirst(HttpHeaders.AUTHORIZATION)

                if (StringUtils.isEmpty(authorization) || !tokenCache.containsKey(authorization)) {
                    return Mono.empty();
                }
                return Mono.just(tokenCache.get(authorization));
            }
        };
    }

    @Bean
    SecurityWebFilterChain springWebFilterChain(ServerHttpSecurity http) {
        return http
                .csrf().disable()
                .formLogin().disable()
                .httpBasic().disable()
                .addFilterAt(authenticationWebFilter(), SecurityWebFiltersOrder.AUTHENTICATION)
                .authorizeExchange()
                .pathMatchers(AUTH_WHITELIST).permitAll()
                .anyExchange().authenticated()
                .and().build();
    }

    //修改为访问数据库的UserDetailsService即可
    @Bean
    ReactiveUserDetailsService userDetailsService() {
        User.UserBuilder userBuilder = User.withDefaultPasswordEncoder();
        UserDetails rob = userBuilder.username("rob").password("rob").roles("USER").build();
        UserDetails admin = userBuilder.username("admin").password("admin").roles("USER", "A
MIN").build();
        return new MapReactiveUserDetailsService(rob, admin);
    }
```

```
    @Bean
    ServerAuthenticationConverter serverAuthenticationConverter() {
        final AnonymousAuthenticationToken anonymous = new AnonymousAuthenticationTok
n("key", "anonymous", AuthorityUtils.createAuthorityList("ROLE_ANONYMOUS"));
        return exchange -> {
            String token = exchange.getRequest().getHeaders().getFirst(HttpHeaders.AUTHORIZAT
ON);
            if (StringUtils.isEmpty(token)) {
                return Mono.just(anonymous);
            }
            if (!token.startsWith(BEARER) || token.length() <= BEARER.length() || !tokenCache.conta
nsKey(token.substring(BEARER.length()))) {
                return Mono.just(anonymous);
            }
            return Mono.just(tokenCache.get(token.substring(BEARER.length())).getAuthentication
));
        };
    }

    @Bean
    AuthenticationWebFilter authenticationWebFilter() {
        AuthenticationWebFilter authenticationWebFilter = new AuthenticationWebFilter(reactiv
AuthenticationManager());

        NegatedServerWebExchangeMatcher negateWhiteList = new NegatedServerWebExchan
eMatcher(ServerWebExchangeMatchers.pathMatchers(AUTH_WHITELIST));
        authenticationWebFilter.setRequiresAuthenticationMatcher(negateWhiteList);

        authenticationWebFilter.setServerAuthenticationConverter(serverAuthenticationConverte
());
        authenticationWebFilter.setSecurityContextRepository(serverSecurityContextRepository())


        authenticationWebFilter.setAuthenticationFailureHandler((webFilterExchange, exception)
-> Mono.error(new BadCredentialsException("权限不足")));
        return authenticationWebFilter;
    }

}
```

login api

```
@RestController
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class LoginController {

    private final ReactiveAuthenticationManager authenticationManager;
    private final ServerSecurityContextRepository contextRepository;

    @PostMapping("/login")
    public Mono login(@RequestBody Login login, ServerWebExchange exchange) {
        final String token = UUID.randomUUID().toString().replaceAll("-", "");
        TokenAuthentication authenticationToken = new TokenAuthentication(token, login.getU
ername(), login.getPassword());
```

```
        return authenticationManager.authenticate(authenticationToken).doOnSuccess(auth ->
ontextRepository.save(exchange, new SecurityContextImpl(authenticationToken))).then(Mono.
ust(token));
    }
}
```

原文链接：Spring WebFlux REST API Token 登陆