MỤC TIÊU

- Tạo và validate JWT phía server
- Sử dụng JWT để truy xuất các tài nguyên được phân quyền

BÀI 1: (2 ĐIỂM)

Tạo REST API có tên PolyApi theo yêu cầu sau

URL	RETURN
GET(/poly/url0)	{url:"/poly/url0", method:"method0()"}
GET(/poly/url1)	{url:"/poly/url1", method:"method1()"}
GET(/poly/url2)	{url:"/poly/url2", method:"method2()"}
GET(/poly/url3)	{url:"/poly/url3", method:"method3()"}
GET(/poly/url4)	{url:"/poly/url4", method:"method4()"}

Kiểm thử REST API với Postman

Hướng dẫn thực hiện:

```
@RestController
public class MyRestApi {

	@GetMapping({"/poly/url0", "/"})
	public Object method0() {
		return Map.of("url", "/poly/url0", "method", "method0()");
	}

	@GetMapping("/poly/url1")
	public Object method1() {
		return Map.of("url", "/poly/url1", "method", "method1()");
	}

	@GetMapping("/poly/url2")
	public Object method2() {
		return Map.of("url", "/poly/url2", "method", "method2()");
	}

	@GetMapping("/poly/url3")
	public Object method3() {
```

```
        return Map.of("url", "/poly/url3", "method", "method3()");
    }

    @GetMapping("/poly/url4")
    public Object method4() {
        return Map.of("url", "/poly/url4", "method", "method4()");
    }
}
```

BÀI 2: (2 ĐIỂM)

- Tạo Spring Bean JwtService cung cấp các phương thức tạo (create) và kiểm tra (validate) JWT

```
@Service
public class JwtService {
    /**
     * Tạo JWT (JSON Web Token)
     *
     * @param user là UserDetails chứa thông tin để tạo token
     * @param expirySeconds là thời hạn có hiệu lực (tính bằng giây)
     */
    public String create(UserDetails user, int expirySeconds) {
        long now = System.currentTimeMillis();
        return Jwts.builder()
                .setClaims(Map.of("name", "Poly"))
                .setSubject(user.getUsername())
                .setIssuedAt(new Date(now))
                .setExpiration(new Date(now + 1000 * expirySeconds))
                .signWith(this.getSigningKey(), SignatureAlgorithm.HS256)
                .compact();
    }
    /**
     * Bóc tách phần body từ JWT
     *
     * @param token là JWT
     */
    public Claims getBody(String token) {
        return Jwts.parserBuilder()
                .setSigningKey(this.getSigningKey()) // xác minh token
                .build().parseClaimsJws(token).getBody();
    }
```

```
        /**
         * Xác minh thời gian hiệu lực
         *
         * @param claims là body của token
         */
        public boolean validate(Claims claims) {
                return claims.getExpiration().after(new Date());
        }

        /**
         * Tạo chữ ký số để ký và xác minh JWT
         */
        poly Key getSigningKey() {
                String secret = "0123456789.0123456789.0123456789"; // HS256 >= 32 byte
                return Keys.hmacShaKeyFor(secret.getBytes());
        }
}
```

- Tạo bộ lọc JwtAuthFilter thực hiện mỗi khi có request đến server thì thực hiện các công việc
  - Bóc lấy JWT từ header Authorization
  - Sử dụng JwtService để kiểm tra JWT
  - Thực hiện đăng nhập tự động (thiết lập Authentication)

```
@Service
public class JwtAuthFilter extends OncePerRequestFilter {
        @Autowired
        JwtService jwtService;

        @Autowired
        UserDetailsService userService;

        @Override
        protected void doFilterInternal(HttpServletRequest request,
                        HttpServletResponse response, FilterChain filterChain)
                        throws ServletException, IOException {
                if (SecurityContextHolder.getContext().getAuthentication() == null) {
                        var authorization = request.getHeader("Authorization");
                        if (authorization != null && authorization.startsWith("Bearer ")) {
                                var token = authorization.substring(7).trim();
                                var claims = jwtService.getBody(token);
                                if (jwtService.validate(claims)) {
                                        var username = claims.getSubject();
```

```
                                        UserDetails user =
userService.loadUserByUsername(username);
                                        var auth = new
UsernamePasswordAuthenticationToken(user, null, user.getAuthorities());

            SecurityContextHolder.getContext().setAuthentication(auth);
                                    }
                                }
                        }
                        filterChain.doFilter(request, response);
            }
}
```

BÀI 3: (2 ĐIỂM)

Sử dụng Spring Boot Security để phân quyền theo yêu cầu sau

- UserDetailsService: Sử dụng InMemoryUserDetailsService với 3 user được mô tả như sau

USERNAME	PASSWORD	ROLES
user@gmail.com	123	USER
admin@gmail.com	123	ADMIN
both@gmail.com	123	ADMIN, USER

- Phân quyền truy xuất theo yêu cầu sau

URL	PERMISSION
/poly/url1	Authenticated
/poly/url2	USER
/poly/url3	ADMIN
/poly/url4	USER, ADMIN
Any Request	PermitAll

Tham khảo mã cấu hình sau

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {
        @Bean
```

```
        public PasswordEncoder passwordEncoder() {
                return PasswordEncoderFactories.createDelegatingPasswordEncoder();
        }

        @Bean
        public UserDetailsService userDetailsService(PasswordEncoder pe) {
                UserDetails user1 =
User.withUsername("user@gmail.com").password(pe.encode("123")).roles("USER").build();
                UserDetails user2 =
User.withUsername("admin@gmail.com").password(pe.encode("123")).roles("ADMIN").build
();
                UserDetails user3 =
User.withUsername("both@gmail.com").password(pe.encode("123")).roles("USER",
"ADMIN").build();
                return new InMemoryUserDetailsManager(user1, user2, user3);
        }

        @Bean
        public SecurityFilterChain securityFilterChain(HttpSecurity http, JwtAuthFilter
jwtAuthFilter) throws Exception {
                http.csrf(config -> config.disable()).cors(config -> config.disable());

                http.authorizeHttpRequests(config -> {
                        config.requestMatchers("/poly/url1").authenticated();
                        config.requestMatchers("/poly/url2").hasRole("USER");
                        config.requestMatchers("/poly/url3").hasRole("ADMIN");
                        config.requestMatchers("/poly/url4").hasAnyRole("USER", "ADMIN");
                        config.anyRequest().permitAll();
                });

                // Bỏ duy trì user trong session
                http.sessionManagement(session -> {
                        session.sessionCreationPolicy(SessionCreationPolicy.STATELESS);
                });

                http.addFilterBefore(jwtAuthFilter,
UsernamePasswordAuthenticationFilter.class);

                return http.build();
        }

        @Bean
```

```
        public AuthenticationManager authenticationManager(AuthenticationConfiguration
config) throws Exception {
                return config.getAuthenticationManager();
        }
}
```

Kiểm thử REST API với Postman

BÀI 4: (2 ĐIỂM)

Tạo LoginController để điều khiển hoạt động đăng nhập với thông tin username và password và trả về JWT

```
@RestController
public class LoginController {
        @Autowired
        AuthenticationManager authenticationManager;

        @Autowired
        JwtService jwtService;

        @PostMapping("/poly/login")
        public Object login(@RequestBody Map<String, String> userInfo) {
                String username = userInfo.get("username");
                String password = userInfo.get("password");
                var authInfo = new UsernamePasswordAuthenticationToken(username,
password);
                var authentication = authenticationManager.authenticate(authInfo);

                if (authentication.isAuthenticated()) {
                        UserDetails user = (UserDetails) authentication.getPrincipal();
                        String token = jwtService.create(user, 20 * 60); // token có hiệu lực 20
phút
                        return Map.of("token", token);
                }
                throw new UsernameNotFoundException("Username not found!");
        }
}
```

Sử dụng Postman để đăng nhập và truy xuất các tài nguyên đã được phân quyền

BÀI 5 (2 ĐIỂM)

Sử dụng Axios để đăng nhập và truy xuất các tài nguyên đã được phân quyền

```
<script>
        let token;
```

```
function login() {
	var url = "http://localhost:8080/poly/login";
	var data = {
		"username": "user@gmail.com",
		"password": "123"
	}
	axios.post(url, data).then(resp => {
		console.log("post() success", resp.data);
		token = resp.data.token;
	}).catch(error => {
		console.log("post() error", error);
	})
}
function gotoUrl0() {
	var url = `http://localhost:8080/poly/url0`;
	axios.get(url).then(resp => {
		console.log("get() success", resp.data);
	}).catch(error => {
		console.log("get() error", error);
	})
}
function gotoUrl1() {
	var url = `http://localhost:8080/poly/url1`;
	var config = {
		headers: {
			"Authorization": `Bearer ${token}`
		}
	};
	axios.get(url, config).then(resp => {
		console.log("get() success", resp.data);
	}).catch(error => {
		console.log("get() error", error);
	})
}
function gotoUrl3() {
	var url = `http://localhost:8080/poly/url3`;
	var config = {
		headers: {
			"Authorization": `Bearer ${token}`
		}
	};
	axios.get(url, config).then(resp => {
```

```
                    console.log("get() success", resp.data);
                }).catch(error => {
                    console.log("get() error", error);
                })
        }
</script>
```