Resource Software Solution

Session 05: Spring Security + JWT

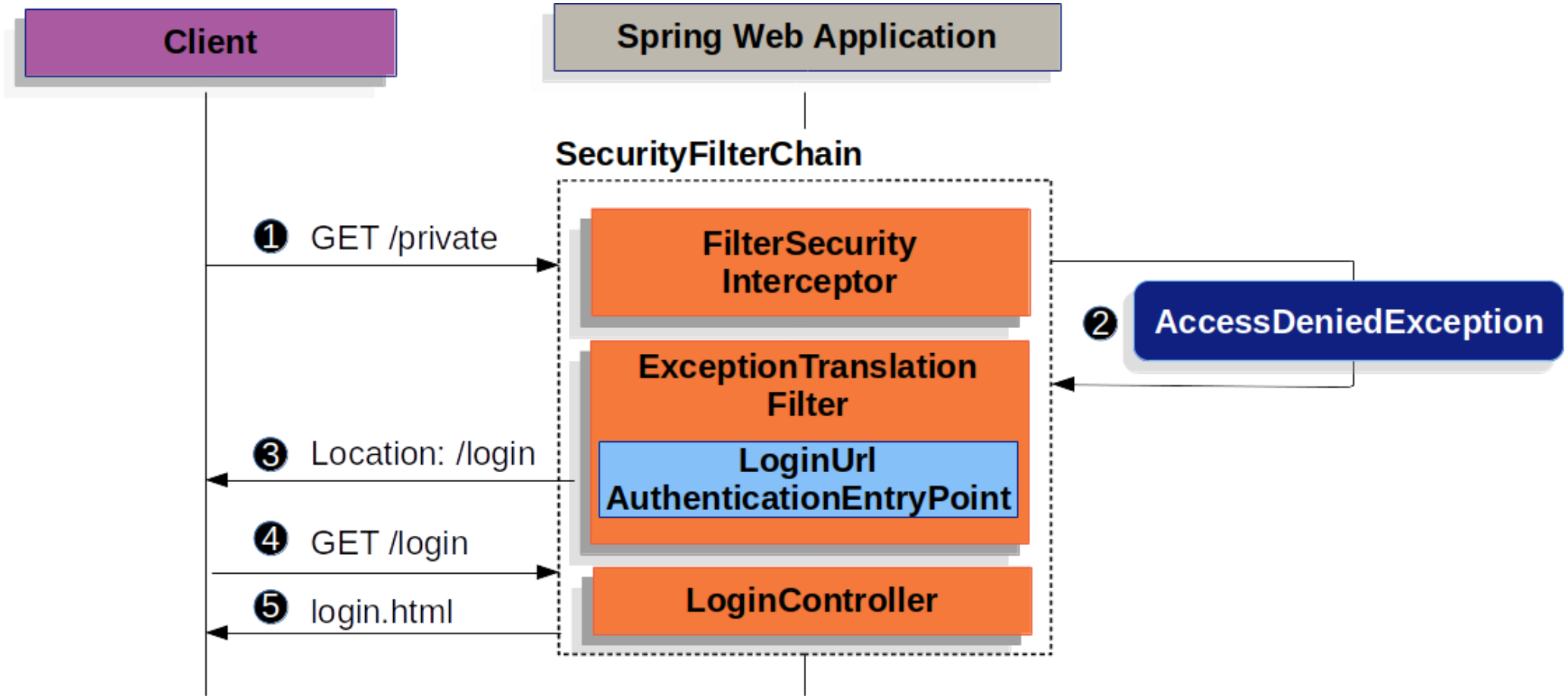
Learning goal

- Spring Security Introduction
- JWT introduction
- How it works
- Demo

Introduction

- Spring Security is a powerful and highly customizable authentication and access-control framework
- Spring Security is a framework that focuses on providing both authentication and authorization to Java applications. Like all Spring projects, the real power of Spring Security is found in how easily it can be extended to meet custom requirements

Introduction



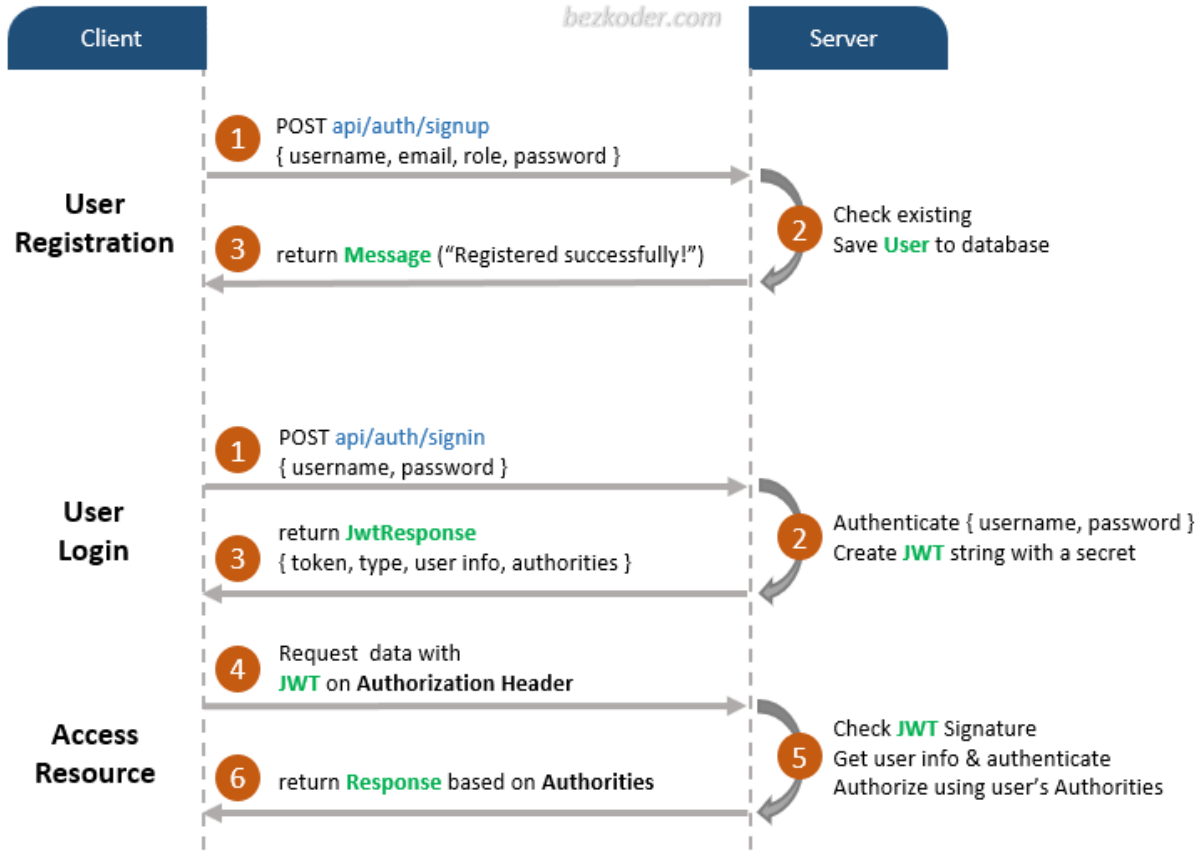
JWT introduction

- JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA.

JWT introduction

Resource Software Solution

How it works?


bezkoder.com
Client
Server
User Registration
1
POST api/auth/signup
{ username, email, role, password }
2
Check existing
Save User to database
3
return Message ("Registered successfully!")
User Login
1
POST api/auth/signin
{ username, password }
2
Authenticate { username, password }
Create JWT string with a secret
3
return JwtResponse
{ token, type, user info, authorities }
Access Resource
4
Request data with
JWT on Authorization Header
5
Check JWT Signature
Get user info & authenticate
Authorize using user's Authorities
6
return Response based on Authorities

Demo - Development Steps

- Step 1: Add dependencies:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-api</artifactId>
    <version>0.11.5</version>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-impl</artifactId>
    <version>0.11.5</version>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-jackson</artifactId>
    <version>0.11.5</version>
</dependency>
```

Demo - Development Steps

- Step 2: Add Configurations:

```
@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
    return http.csrf().disable() HttpSecurity
            .authorizeHttpRequests() AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestM
            .requestMatchers( ...patterns: "/register","/authenticate").permitAll()
            .and() HttpSecurity
            .authorizeHttpRequests().requestMatchers( ...patterns: "/items/**") AuthorizeHttpRe
            .authenticated().and() HttpSecurity
            .sessionManagement() SessionManagementConfigurer<HttpSecurity>
            .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
            .and() HttpSecurity
            .authenticationProvider(authenticationProvider())
            .addFilterBefore(authFilter, UsernamePasswordAuthenticationFilter.class)
            .build();
}
```

Demo - Development Steps

- Step 3: Create filter:

```java
@Component
public class JwtAuthFilter extends OncePerRequestFilter {

    @Autowired
    private JwtUtil jwtUtil;

    @Autowired
    private UserInfoUserDetailsService userDetailsService;

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
            throws ServletException, IOException {
        String authHeader = request.getHeader( s: "Authorization");
        String token = null;
        String username = null;
        if (authHeader != null && authHeader.startsWith("Bearer ")) {
            token = authHeader.substring( beginIndex: 7);
            username = jwtUtil.extractUsername(token);
        }

        if (username != null && SecurityContextHolder.getContext().getAuthentication() == null) {
            UserDetails userDetails = userDetailsService.loadUserByUsername(username);
            if (jwtUtil.validateToken(token, userDetails)) {
                UsernamePasswordAuthenticationToken authToken =
                        new UsernamePasswordAuthenticationToken(userDetails, credentials: null, userDetails.getAuthorities());
                authToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));
                SecurityContextHolder.getContext().setAuthentication(authToken);
            }
        }
        filterChain.doFilter(request, response);
    }
}
```

Demo - Development Steps

- Step 4: Some functions in JwtUtil:

```
private Boolean isTokenExpired(String token) {
    return extractExpiration(token).before(new Date());
}

public Boolean validateToken(String token, UserDetails userDetails) {
    final String username = extractUsername(token);
    return (username.equals(userDetails.getUsername()) && !isTokenExpired(token));
}


public String generateToken(String userName){
    Map<String,Object> claims=new HashMap<>();
    return createToken(claims,userName);
}
```

Demo - Development Steps

- Step 5: APIs for registration and generate jwt:

```
@PostMapping("/register")
public String addNewUser(@RequestBody User user) {
    return userService.insert(user);
}

@PostMapping("/authenticate")
public String authenticateAndGetToken(@RequestBody AuthRequest authRequest) {
    Authentication authentication = authenticationManager
            .authenticate(new UsernamePasswordAuthenticationToken(
                    authRequest.getUsername(), authRequest.getPassword()));
    if (authentication.isAuthenticated()) {
        return jwtUtil.generateToken(authRequest.getUsername());
    } else {
        throw new UsernameNotFoundException("User name not found");
    }
}
```

Demo - Development Steps

- Step 6: APIs for testing:

```
@GetMapping("/welcome")
public String welcome() {
    return "Welcome to homepage";
}

@GetMapping
@PreAuthorize("hasAuthority('ADMIN')")
public List<Product> getAll() {
    return userService.getItems();
}

@GetMapping("/{id}")
@PreAuthorize("hasAuthority('USER')")
public Product getItemById(@PathVariable int id) { return userService.getProduct(id); }
```

*Keeping up those **inspiration** and the **enthusiasm** in the **learning path**.*
*Let confidence to bring it into **your career path** for getting gain the **success** as your expectation.*

Thank you

Contact

- Name: R2S Academy

- Email: daotao@r2s.edu.vn

- Hotline/Zalo: 0919 365 363

- Website: https://r2s.edu.vn

- Fanpage: https://www.facebook.com/r2s.tuyendung

Questions and Answers