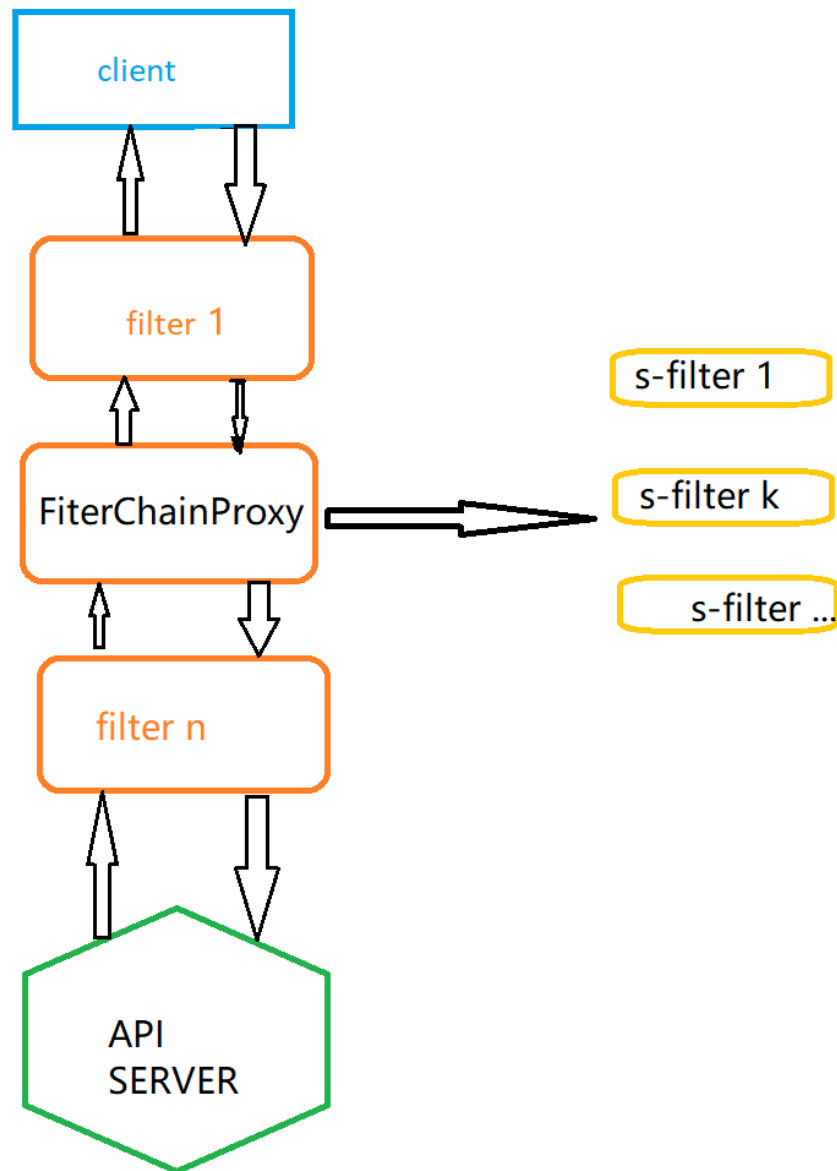


# Spring Security + JWT

## Authentication work flow



in `Servlet` filters chain, Spring Security add `FilterChainProxy`. `FilterChainProxy`:

```
1 @Override
2 public void doFilter(ServletRequest request, ServletResponse response,
3                     FilterChain chain) throws IOException, ServletException
4 {
5     ...
6     // get Spring Security filters
7     List<Filter> filters = getFilters(request);
8     // Spring Security
9     VirtualFilterChain vfc = new VirtualFilterChain(fwRequest, chain,
10 filters);
```

```

10     vfc.doFilter(request, response);
11
12     ...
13 }
14

```

the filters:

```

v {..} filters = {ArrayList@6858} size = 15
  > {..} 0 = {WebAsyncManagerIntegrationFilter@6863}
  > {..} 1 = {SecurityContextPersistenceFilter@6864}
  > {..} 2 = {HeaderWriterFilter@6865}
  > {..} 3 = {CsrfFilter@6866}
  > {..} 4 = {LogoutFilter@6867}
  > {..} 5 = {UsernamePasswordAuthenticationFilter@6868}
  > {..} 6 = {DefaultLoginPageGeneratingFilter@6869}
  > {..} 7 = {DefaultLogoutPageGeneratingFilter@6870}
  > {..} 8 = {BasicAuthenticationFilter@6871}
  > {..} 9 = {RequestCacheAwareFilter@6872}
  > {..} 10 = {SecurityContextHolderAwareRequestFilter@6873}
  > {..} 11 = {AnonymousAuthenticationFilter@6874}
  > {..} 12 = {SessionManagementFilter@6875}
  > {..} 13 = {ExceptionTranslationFilter@6876}
  > {..} 14 = {FilterSecurityInterceptor@6877}

```

`UsernamePasswordAuthenticationFilter` for Authentication

`FilterSecurityInterceptor` for Authorization.

```

1 <dependency>
2   <groupId>org.springframework.boot</groupId>
3   <artifactId>spring-boot-starter-security</artifactId>
4 </dependency>
5

```

```

1 @EnableWebSecurity
2 public class SecurityConfig extends WebSecurityConfigurerAdapter {
3 }
4

```

💡 `SecurityContextHolder` manager it and you can use it everywhere:

```

1 Authentication authentication =
  SecurityContextHolder.getContext().getAuthentication();
2

```

the path: `SecurityContextHolder` → `SecurityContext` → `Authentication`。

📝 `Authentication`: save the user info

📝 `SecurityContext`: for get `Authentication`

📝 `SecurityContextHolder`: for get `SecurityContext`

the relationship:



`Authentication`:

📝 `Principal`: before authentication is user name, after is user object.

📝 `Credentials`: usually is password

📝 `Authorities`: permissions

sql the user info from database → check password → true save it to `SecurityContext` → `SecurityContext` contains `authentication` means user login success

Spring Security:

```

1 Authentication authentication = new
  UsernamePasswordAuthenticationToken(userName, password, permissions);
2 SecurityContextHolder.getContext().setAuthentication(authentication);
3

```

the path: `Authentication` → `SecurityContext` → `SecurityContextHolder`。

```

1 // check user
2 if (!userService.login(userName, password)) {
3     return "User or Password not match";
4 }
5 //
6 Authentication authentication = new
UsernamePasswordAuthenticationToken(userName, password, permissions);
7 SecurityContextHolder.getContext().setAuthentication(authentication);
8

```

Spring security using the  `AuthenticationManager` for Authentication

```

1 @EnableWebSecurity
2 public class SecurityConfig extends WebSecurityConfigurerAdapter {
3     @Bean
4     @Override
5     protected AuthenticationManager authenticationManager() throws Exception
6     {
7         return super.authenticationManager();
8     }
9 }
10

```

```

1 @RestController
2 @RequestMapping("/API")
3 public class LoginController {
4     @Autowired
5     private AuthenticationManager authenticationManager;
6
7     @PostMapping("/login")
8     public String login(@RequestBody LoginParam param) {
9         //create token
10         Authentication token = new
UsernamePasswordAuthenticationToken(param.getUsername(),
param.getPassword());
11         // AuthenticationManager verify the token
12         Authentication authentication =
authenticationManager.authenticate(token);
13         // save it to context
14         SecurityContextHolder.getContext().setAuthentication(authentication);
15         return "success";
16     }
17 }
18

```

Session `HttpSessionSecurityContextRepository.SPRING_SECURITY_CONTEXT_KEY` you can get Authentication:

```
1 Authentication =  
  (Authentication)session.getAttribute(HttpSessionSecurityContextRepository.SPRING_SECURITY_CONTEXT_KEY)  
2
```

:

```
1 @GetMapping("/logout")  
2 public String logout() {  
3     SecurityContextHolder.clearContext();  
4     return "logout";  
5 }  
6
```

## JWT

disable the session:

```
1 //  
2 http.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);  
3
```

|

create token:

```
1 @Autowired  
2 private UserService userService;  
3  
4 @PostMapping("/login")  
5 public UserVO login(@RequestBody @Validated LoginParam user) {  
6     // call userService  
7     return userService.login(user);  
8 }  
9
```

```
1 public UserVO login(LoginParam param) {  
2     //  
3     UserEntity user = userMapper.selectByUsername(param.getUsername());  
4     //  
5     if (user == null || !passwordEncoder.matches(param.getPassword(),  
6         user.getPassword())) {  
7         throw new ApiException("Error");  
8     }  
9     //
```

```

10     UserVO userVO = new UserVO();
11     userVO.setId(user.getId())
12         .setUsername(user.getUsername())
13         //
14         .setToken(jwtManager.generate(user.getUsername()));
15     return userVO;
16 }
17

```

add filter:

```

1  @Component
2  public class LoginFilter extends OncePerRequestFilter {
3      @Autowired
4      private JwtManager jwtManager;
5      @Autowired
6      private UserServiceImpl userService;
7
8      @Override
9      protected void doFilterInternal(HttpServletRequest request,
10                                     HttpServletResponse response, FilterChain chain) throws ServletException,
11                                     IOException {
12
13         Claims claims =
14             jwtManager.parse(request.getHeader("Authorization"));
15         if (claims != null) {
16
17             String username = claims.getSubject();
18
19             UserDetails user = userService.loadUserByUsername(username);
20
21             Authentication authentication = new
22                 UsernamePasswordAuthenticationToken(user, user.getPassword(),
23                 user.getAuthorities());
24
25             SecurityContextHolder.getContext().setAuthentication(authentication);
26         }
27         chain.doFilter(request, response);
28     }
29 }
30

```

replace the filter in securityConfig :

```

1
2  http.addFilterBefore(loginFilter,
3  UsernamePasswordAuthenticationFilter.class);
4

```

login :

Postman

File Edit View Help

+ New Import Runner

My Workspace Invite

Filter

History Collections APIs

+ New Collection Trash

security 3 requests

vueback 3 requests

POST Login GET Get user list POST Create user

Examples 0 BUILD

POST http://localhost:8091/API/login Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1
2 "username": "admin",
3 "password": "12345"
4
```

Body Cookies Headers (13) Test Results

Status: 200 OK Time: 1697 ms Size: 715 B Save Response

Pretty Raw Preview Visualize JSON

```
1
2 {
3   "code": 0,
4   "msg": "Success",
5   "data": {
6     "id": 1,
7     "username": "admin",
8     "token": "eyJhbGciOiJIUzUxMi9.eyJ3ZGVlI0IjZG1pb1IsIm1hdCI6MTYwOTg2NjAyIiwiaXNjaS0tYyNDIjFQ.
9       CSXGhH1veCSKX9s6q7DX1U98cqdmF4zHn3OMU_pFwh8TTvPKXdeb8CieV7IDMdhHt917vZ1y1J_lqRpmITc0A",
10    "resourceIds": [
11      1,
12      2001,
13      2,
14      2002,
15      3,
16      2003,
17      1001,
18      1002,
19      1003,
20      1004
21    ]
22  }
23 }
```

Find and Replace Console

Bootcamp Build Browse

## get user list

Postman

File Edit View Help

+ New Import Runner

My Workspace Invite

Filter

History Collections APIs

+ New Collection Trash

security 3 requests

vueback 3 requests

POST Login GET Get user list POST Create user

Examples 0 BUILD

GET http://localhost:8091/API/user/page/1 Send Save

Params Authorization Headers (7) Body Pre-request Script Tests Settings

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Postman-Token	<calculated when request is sent>	
<input checked="" type="checkbox"/> Host	<calculated when request is sent>	
<input checked="" type="checkbox"/> User-Agent	PostmanRuntime/7.26.8	
<input checked="" type="checkbox"/> Accept	*/*	
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br	
<input checked="" type="checkbox"/> Connection	keep-alive	
<input checked="" type="checkbox"/> Authorization	eyJhbGciOiJIUzUxMi9.eyJ3ZGVlI0IjZG1pb1IsIm1hdCI6MTYwOTg2NjAyIiwiaXNjaS0tYyNDIjFQ. CSXGhH1veCSKX9s6q7DX1U98cqdmF4zHn3OMU_pFwh8TTvPKXdeb8CieV7IDMdhHt917vZ1y1J_lqRpmITc0A	
Key	Value	Description

Body Cookies Headers (13) Test Results

Status: 200 OK Time: 462 ms Size: 869 B Save Response

Pretty Raw Preview Visualize JSON

```
1
2 {
3   "code": 0,
4   "msg": "Success",
5   "data": {
6     "records": [
7       {
8         "id": 2,
9         "username": "user1",
10        "roleIds": [
11          2
12        ],
13        "companyIds": [
14          2
15        ]
16      }
17     ]
18   }
19 }
```

Find and Replace Console

Bootcamp Build Browse

