



Autorização por role no controller - Spring Security

A autorização para acesso a métodos através da **role** pode ser feita a nível de controller, ou seja, nem chegaríamos na camada de segurança para validar se um determinado recurso pode ou não ser acessado por um usuário com determinada role.

Para fazer essa configuração é necessário **habilitar uma anotação na classe de segurança**, passando um **parâmetro que diz que os filtros de acesso serão feitos pelo controller**:

```
@Configuration
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private UserDetailsServiceImpl userDetailsService;

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .httpBasic().httpBasicConfigurer(HttpSecurityConfigurerAdapter::httpBasic)
            .and().httpSecurity()
            .authorizeHttpRequests().authorizeHttpRequests(request -> {
                requestMatchers(HttpMethod.GET, "/parking-spot/**").permitAll();
                requestMatchers(HttpMethod.POST, "/parking-spot/**").hasRole(RoleName.ROLE_USER.name());
                requestMatchers(HttpMethod.DELETE, "/parking-spot/**").hasRole(RoleName.ROLE_ADMIN.name());
            })
            .and().httpSecurity()
            .csrf().disable();
    }
}
```

```
@Configuration
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private UserDetailsServiceImpl userDetailsService;

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .httpBasic().httpBasicConfigurer(HttpSecurityConfigurerAdapter::httpBasic)
            .and().httpSecurity()
            .authorizeHttpRequests().authorizeHttpRequests(request -> {
                requestMatchers(HttpMethod.GET, "/parking-spot/**").permitAll();
                requestMatchers(HttpMethod.POST, "/parking-spot/**").hasRole(RoleName.ROLE_USER.name());
                requestMatchers(HttpMethod.DELETE, "/parking-spot/**").hasRole(RoleName.ROLE_ADMIN.name());
            })
            .and().httpSecurity()
            .csrf().disable();
    }
}
```

Agora no **controller** basta eu anotar os endpoints com um **@PreAuthorize** passando as roles que podem acessa-los:

```
@PreAuthorize("hasRole('ROLE_ADMIN')")
@PostMapping
public ResponseEntity<Object> saveParkingSpot(@RequestBody @Valid ParkingSpotDto parkingSpotDto) {
    if (parkingSpotService.existsByLicensePlateCar(parkingSpotDto.getLicensePlateCar())) {
        return ResponseEntity.status(HttpStatus.CONFLICT).body("Conflict: License Plate Car is already in use!");
    }
    if (parkingSpotService.existsByParkingSpotNumber(parkingSpotDto.getParkingSpotNumber())) {
        return ResponseEntity.status(HttpStatus.CONFLICT).body("Conflict: Parking Spot is already in use!");
    }
    if (parkingSpotService.existsByApartmentAndBlock(parkingSpotDto.getApartment(), parkingSpotDto.getBlock())) {
        return ResponseEntity.status(HttpStatus.CONFLICT).body("Conflict: Parking Spot already registered for this apartment/block!");
    }
    var parkingSpotModel = new ParkingSpotModel();
    BeanUtils.copyProperties(parkingSpotDto, parkingSpotModel);
    parkingSpotModel.setRegistrationDate(LocalDateTime.now(ZoneId.of("UTC")));
    return ResponseEntity.status(HttpStatus.CREATED).body(parkingSpotService.save(parkingSpotModel));
}
```

```
@PreAuthorize("hasAnyRole('ROLE_ADMIN', 'ROLE_USER')")
@GetMapping
public ResponseEntity<Page<ParkingSpotModel>> getAllParkingSpots() {
    return ResponseEntity.status(HttpStatus.OK).body(parkingSpotService.findAll());
}
```