הראשות בעזרת authority

נניח שאנחנו רוצים ליצור admin מתלמד שיהיו לו רק אפשרויות קריאה

אז הוספנו role חדש בenum שלנו שיש לו רק אפשרויות קריאה

```
ADMIN_TRAINEE(Sets.newHashSet(COURSE_READ,STUDENT_READ));
```

וזה היוזר שיצרנו

```
UserDetails traineeAdmin = User.builder()
        .username("traineeAdmin")
        .password(passwordEncoder.encode( rawPassword: "password"))
        .roles(ADMIN_TRAINEE.name())
        .build();
```

יצרנו controller חדש

```
@RestController
@RequestMapping("management/api/v1/students")
public class StudentManagementController {

    private static final List<Student> studentList = List.of(
            new Student( studentId: 1,  studentName: "James Bond"),
            new Student( studentId: 2,  studentName: "Shay Gali"),
            new Student( studentId: 3,  studentName: "Anna Smith")
    );

    @GetMapping
    public List<Student> getStudentList(){
        return studentList;
    }

    @PostMapping
    public void registerNewStudent(@RequestBody Student student){
        System.out.println(student);
    }

    @DeleteMapping("/{id}")
    public void deleteStudent(@PathVariable("id") Integer studentId){
        System.out.println(studentId);
    }

    @PutMapping("/{id}")
    public void updateStudent(@PathVariable("id") Integer studentId,@RequestBody Student student){
        System.out.printf("%s %s%n",studentId,student);
    }
}
```

עכשיו אם ננסה לעשות get נצליח אבל post הוא לא יעבוד

בהמשך הוא יסביר למה זה קורה – אבל זה מה שאמור לקרות

http://localhost:8080/management/api/v1/students

POST http://localhost:8080/management/api/v1/students

Params Authorization Headers (10) Body Pre-request Script Tests Settings

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

```
1 {
2     "studentId": 3,
3     "studentName": "Anna Smith"
4 }
```

Body Cookies (1) Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "timestamp": "2022-04-19T13:13:59.295+00:00",
3     "status": 403,
4     "error": "Forbidden",
5     "path": "/management/api/v1/students"
6 }
```

עכשיו כדי שזה יעבוד נעשה את זה (הסבר בהמשך)

```
@Override
    protected void configure(HttpSecurity http) throws Exception {
    http
            .csrf().disable()
            .authorizeRequests()// אנחנו רוצים לאמת בקשות
```

יש שתי דרכים שנוכל להגביל לפי ההרשאות: במתודה של configure ועם אנוטציות.

כאן ציינו בדיוק עם איזה הרשאה אפשר לגשת למתודות delet,post,put ולget איזה role

```
.antMatchers(HttpMethod.DELETE, ...antPatterns: "/management/api/**").hasAnyAuthority(COURSE_WRITE.name())
.antMatchers(HttpMethod.POST, ...antPatterns: "/management/api/**").hasAnyAuthority(COURSE_WRITE.name())
.antMatchers(HttpMethod.PUT, ...antPatterns: "/management/api/**").hasAnyAuthority(COURSE_WRITE.name())
.antMatchers(HttpMethod.GET, ...antPatterns: "/management/api/**").hasAnyRole(ADMIN.name(),ADMIN_TRAINEE.name())
```

וכדי להגדיר את הauthority ניצור מתודה חדשה בenum של הrole, משהיא בעצם עושה זה להוסיף את כל ההרשאות שיש לאותו role וסט של SimpleGrantedAuthority, ובסוף היא תיצור אחד חדש עם השם של הrole כשבהתחלה יש _ROLE(כי ככה זה יהיה אם היינו משתמשים ב role ביוזר)

```
public Set<SimpleGrantedAuthority> getGrantedAuthorities() {
    Set<SimpleGrantedAuthority> permissions = getPermissions().stream()
            .map(permission ->
                    new SimpleGrantedAuthority(permission.getPermission()))
            .collect(Collectors.toSet());
    permissions.add(new SimpleGrantedAuthority( role: "ROLE_" + this.name()));
    return permissions;
}
```

ואז נעדכן במקום role ל authorities

```
@Bean
protected UserDetailsService userDetailsService() {
    UserDetails rootUser = User.builder()
            .username("admin")
            .password(passwordEncoder.encode( rawPassword: "password"))
              .roles(ADMIN.name()) ...
            .authorities(ADMIN.getGrantedAuthorities())
            .build();

    UserDetails studentUser = User.builder()
            .username("student")
            .password(passwordEncoder.encode( rawPassword: "password"))
              .roles(STUDENT.name())
            .authorities(STUDENT.getGrantedAuthorities())

            .build();

    UserDetails traineeAdmin = User.builder()
            .username("traineeAdmin")
            .password(passwordEncoder.encode( rawPassword: "password"))
              .roles(ADMIN_TRAINEE.name())
            .authorities(ADMIN_TRAINEE.getGrantedAuthorities())

            .build();

    return new InMemoryUserDetailsManager(rootUser, studentUser,traineeAdmin);
}
```

אם נדבג ונראה מה בעצם מוחזר מהפונקציה אפשר לראות איזה הרשאות יש לנו פה

```
rootUser = {User@5590} "org.springframework.security.core.userdetails.User [Username=admin, Password=[PROTECTED], Enabled=true, AccountNon... View
  password = "$2a$10$qeChhXoaH1Edh.yn1LV5nuqNB6OKQOkqpKVjYRYoNMMKPWxrhyfii"
  username = "admin"
  authorities = {Collections$UnmodifiableSet@6335}  size = 5
    0 = {SimpleGrantedAuthority@6337} "ROLE_ADMIN"
    1 = {SimpleGrantedAuthority@6338} "course:read"
    2 = {SimpleGrantedAuthority@6339} "course:write"
    3 = {SimpleGrantedAuthority@6340} "student:read"
    4 = {SimpleGrantedAuthority@6341} "student:write"
  accountNonExpired = true
  accountNonLocked = true
  credentialsNonExpired = true
  enabled = true
```

```
studentUser = {User@5591} "org.springframework.security.core.userdetails.User [Username=student, Password=[PROTECTED], Enabled=true, Accoun... View
  password = "$2a$10$Rdf/8KqD8uYue4yPrjV4E.MIcenv/0b8hYkflPn3UazLWjOKyTt2m"
  username = "student"
  authorities = {Collections$UnmodifiableSet@6346}  size = 1
    0 = {SimpleGrantedAuthority@6348} "ROLE_STUDENT"
  accountNonExpired = true
  accountNonLocked = true
  credentialsNonExpired = true
  enabled = true
```

```
traineeAdmin = {User@5592} "org.springframework.security.core.userdetails.User [Username=traineeAdmin, Password=[PROTECTED], Enabled=true, ... View
  password = "$2a$10$QYblvZW0gxDL9inIgSRILu171M.KCtYBmBejbD8qdypy/7p8HzA4u"
  username = "traineeAdmin"
  authorities = {Collections$UnmodifiableSet@6357}  size = 3
    0 = {SimpleGrantedAuthority@6359} "ROLE_ADMIN_TRAINEE"
    1 = {SimpleGrantedAuthority@6360} "course:read"
    2 = {SimpleGrantedAuthority@6361} "student:read"
  accountNonExpired = true
  accountNonLocked = true
  credentialsNonExpired = true
  enabled = true
```

אפשר לראות מה קורה אם משתמשים ב authorities מה שקורה

```
public User.UserBuilder authorities(GrantedAuthority... authorities) {
    return this.authorities((Collection)Arrays.asList(authorities));
}

public User.UserBuilder authorities(Collection<? extends GrantedAuthority> authorities) {
    this.authorities = new ArrayList(authorities);
    return this;
}

public User.UserBuilder authorities(String... authorities) {
    return this.authorities((Collection)AuthorityUtils.createAuthorityList(authorities));
}
```

וב role -> הוא בעצם ישים פשוט את הroles שהכנסנו לו לרשימה של SimpleGranted עם תחילית של _ROLE, ואז ישתמש ב authorities

```
public User.UserBuilder roles(String... roles) {
    List<GrantedAuthority> authorities = new ArrayList(roles.length);
    String[] var3 = roles;
    int var4 = roles.length;

    for(int var5 = 0; var5 < var4; ++var5) {
        String role = var3[var5];
        Assert.isTrue(!role.startsWith("ROLE_"), () -> {
            return role + " cannot start with ROLE_ (it is automatically added)";
        });
        authorities.add(new SimpleGrantedAuthority( role: "ROLE_" + role));
    }

    return this.authorities((Collection)authorities);
}
```

ולמעלה נשנה את זה מname ל permission

```
.antMatchers(HttpMethod.DELETE, ...antPatterns: "/management/api/**").hasAnyAuthority(COURSE_WRITE.getPermission())
.antMatchers(HttpMethod.POST, ...antPatterns: "/management/api/**").hasAnyAuthority(COURSE_WRITE.getPermission())
.antMatchers(HttpMethod.PUT, ...antPatterns: "/management/api/**").hasAnyAuthority(COURSE_WRITE.getPermission())
.antMatchers(HttpMethod.GET, ...antPatterns: "/management/api/**").hasAnyRole(ADMIN.name(),ADMIN_TRAINEE.name())
```

כי זה מה שיהיה לו בעצם (את הלמטה נשאיר name כי זה מה שיהיה שם)

אז במתודה שיצרנו של getGrantedAuthorities זה בעצם מה שעשינו כדי להשתמש בauthorities ולא לחזור על קוד. (בעצם נגיד role ששם יהיו כל ההרשאות ששייכות לאותה הגדרה ומשם אנחנו בעצמנו ניצור את ה authorities שצריך)

דבר חשוב על הantMatchers : איך זה עובד הוא יעבור אחד אחד – האם אני יכול לגשת לפה או האם אני ניגש לפה וברגע שהוא מגיע לכן או לא הוא יעשה את זה – למשל אם נשים את זה ככה ומישהו ניגש ל**/management/api/ ויש לו role או של admin או של admin trainee הוא יוכל לגשת למרות שאחרי זה יש לנו את ההגדרות עם הauthority שלא יסכימו לו לגשת.

לכן הסדר שלהם חשוב, וצריך לשים לב מה שמים לפני מה

```
@Override
    protected void configure(HttpSecurity http) throws Exception {
    http
            .csrf().disable() HttpSecurity
            .authorizeRequests()// אנחנו רוצים לאמת בקשות
            .antMatchers( ...antPatterns: "/", "/index.html", "/css/*", "/js/*").permitAll() // בנתיבים האלה תאשר לכולם
            .antMatchers( ...antPatterns: "/api/**").hasRole(STUDENT.name()) // כל מה שהוא לפי הדפוס
            .antMatchers( ...antPatterns: "/management/api/**").hasAnyRole(ADMIN.name(),ADMIN_TRAINEE.name()) ExpressionUrlAuthorizationConfigurer
            .antMatchers(HttpMethod.DELETE, ...antPatterns: "/management/api/**").hasAnyAuthority(COURSE_WRITE.getPermission())
            .antMatchers(HttpMethod.POST, ...antPatterns: "/management/api/**").hasAnyAuthority(COURSE_WRITE.getPermission())
            .antMatchers(HttpMethod.PUT, ...antPatterns: "/management/api/**").hasAnyAuthority(COURSE_WRITE.getPermission())
            .anyRequest() // חוץ מאלה תאמת את כל השאר
            .authenticated() //עם שם משתמש וסיסמא
            .and() HttpSecurity
            .httpBasic(); // והדרך היא עם
}
```