LU10a - Umgang mit Exceptions

Spring ResponseStatusException

Allgemein gilt, dass Exceptions in Bezug auf die Performanz schlechter abschneiden als die entsprechenden Checks. Daher sollte man Exceptions nie in „Schönwetter“-Fällen benutzen, sondern höchstens in unerwarteten Situationen. Ein ungültiger API-Aufruf ist in unserem Fall aber unerwartet und kann mittels einer Exception gelöst werden. Werden aber viele ungültige API-Aufrufe erwartet, sollte man eine Exception-freie Alternative in Erwägung ziehen.

Spring bietet eine sehr einfache Version, um Fehlerfälle mittels Exceptions abzuhandeln. Dabei wird direkt automatisch eine Antwort mit dem entsprechenden HTTP-Statuscode erstellt und zurückgeschickt.

Lange Variante	Verkürzte Variante	Mit Exception
`Optional<Project> optionalProject = projectRepository.findById(projectName); if(optionalProject.isEmpty()){ return ResponseEntity.notFound().build(); } Project project = optionalProject.get();`	`Project project = projectRepository.findById(projectName).orElse(null); if(null == project){ return ResponseEntity.notFound().build(); }`	`Project project = projectRepository.findById(projectName).orElseThrow( () -> new ResponseStatusException(HttpStatus.NOT_FOUND, "Project not found"));`

Validierung des JWT

In der vorgeschlagenen Klasse `JwtUtil.java` gibt es bereits eine Möglichkeit, um ein JWT zu prüfen (Siehe auch LU08.A05 Register/Login).

```
public boolean validateToken(String token) {
    try {
Jwts.parserBuilder().setSigningKey(key).build().parseClaimsJws(token);
        return true;
    } catch (Exception e) {
        return false;
    }
}
```

Kleine Wiederholung

Setzt man Exceptions ein, sollte man diese auch loggen oder weiterschicken (gegebenenfalls sogar ans Frontend). Ein „catch“ ohne Verwendung der gefangenen Exception, wie im Beispiel, ist zu vermeiden.

Wenn zudem Informationen wie das „Subject“ aus dem JWT herausgelesen werden müssen, ergibt es keinen Sinn, eine separate „Exception“-basierte Validierung zu machen.

```
public String verifyTokenAndExtractSubject() {
    try{
```

```
        String token = extractTokenFromHeader();
        return extractSubject(token);
    } catch (Exception e) {
        // Exception mitschicken
        // throw new ResponseStatusException(HttpStatus.UNAUTHORIZED,
"Invalid token", e);

        // Exception loggen
        log.warn("Invalid token", e);
        throw new ResponseStatusException(HttpStatus.UNAUTHORIZED, "Invalid
token");
    }
}

private String extractSubject(String token) {
    return Jwts.parserBuilder()
            .setSigningKey(key)
            .build()
            .parseClaimsJws(token)
            .getBody()
            .getSubject();
}

private String extractTokenFromHeader() {
    ServletRequestAttributes attributes = (ServletRequestAttributes)
RequestContextHolder.getRequestAttributes();
    if (attributes == null){
        return null;
    }
    String authHeader = attributes.getRequest().getHeader("Authorization");
    if (authHeader == null || !authHeader.startsWith("Bearer ")) {
        return null;
    }
    return authHeader.substring(7);
}
```