Software Engineering 2 Lösungen der Übung

Dr. F.-K. Koschnick, Sybit GmbH

Aufgabe (Story): REST-Service für Retros

Als Schnittstellennutzer möchte ich über einen REST-Service die Retros für ein beliebiges Projekt abgreifen können, um mit den Daten Auswertungen durchführen zu können.

Verifiziere, dass

- mit einem GET-Aufruf der URL /projects/{id}/retros alle Retros zu dem Projekt mit der entsprechenden IDs abgerufen werden können
- das Mediaformat der Daten JSON ist
- der Service nicht mit der Session abgesichert ist

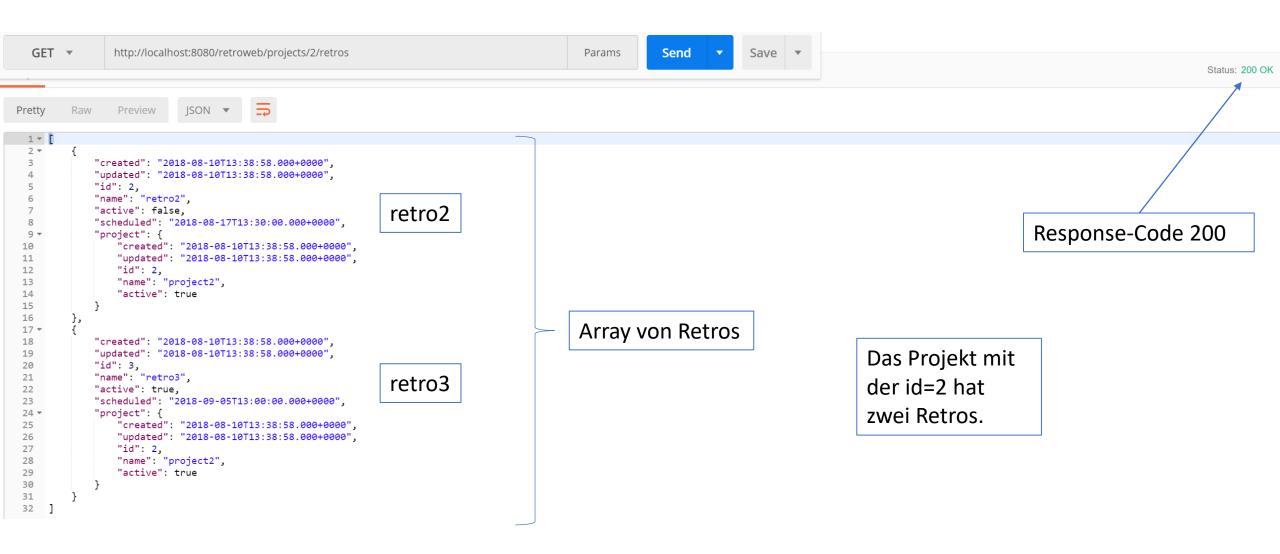
Technische Bemerkung:

Einen REST-Controller mit Spring-Boot bauen und den RetroService nutzen.

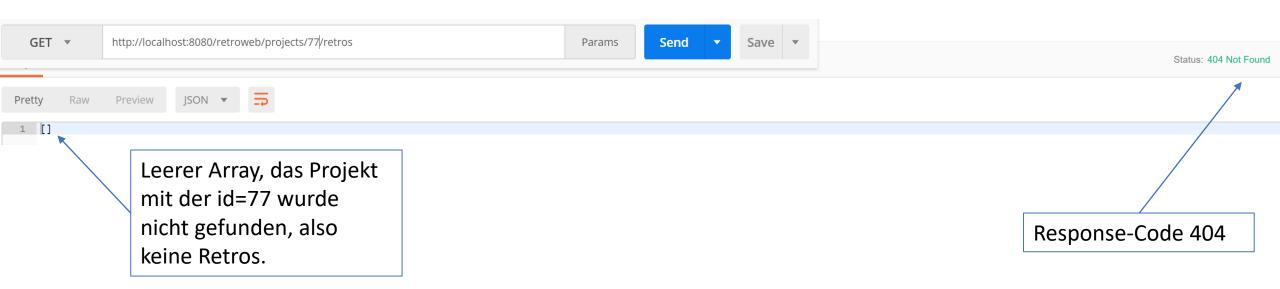
```
import java.util.ArrayList;
                                                                                             Controller
import java.util.List;
import javax.servlet.http.HttpServletResponse;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.PathVariable;
                                                                                      Annotation, dass es sich um einen Rest-Controller handelt!!!
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import de.htwg.retroweb.exception.ResourceNotFoundException;
import de.htwg.retroweb.model.Projects;
import de.htwg.retroweb.model.Retros;
                                                  @RestController
import de.htwg.retroweb.service.ProjectsService;
                                                  public class ExerciseRetrosRestController {
                                                                                                RequestMapping für die REST-Url
import de.htwg.retroweb.service.RetrosService;
                                                    @Autowired
                                                  private RetroService retroService;
                                                                                                                   Es soll JSON zurückgeliefert werden.
                                                    @Autowired
                                                    private ProjectService projectService;
                                                  @RequestMapping(value = "projects/{id}/retros", method = RequestMethod. GET, produces = MediaType. APPLICATION JSON VALUE)
                                                    public List<Retro> getRetrosFromProject(@PathVariable(value = "id") Long projectId, HttpServletResponse response) {
                                                      Project project;
                                                      try {
                                                        project = projectService.getById(projectId);
                                                                                                     Zuerst wird das Projekt geholt, dann mit dem
 Falls das Projekt nicht
                                                        return retroService.findByProject(project);_
                                                                                                     Objekt project werden die Retros für das
                                                      } catch (ResourceNotFoundException e) {
 existiert, wird ein 404
                                                        response.setStatus(404);
                                                                                                     Projekt geholt und zurückgegeben. Die
                                                        return new ArrayList<Retros>();
 zurückgegeben und eine
                                                                                                     Konvertierung nach JSON geht automatisch.
 leere Retro-Liste.
```

package de.htwg.retroweb.controller.rest;

Ergebnis in Postman bei Aufruf: http://localhost:8080/retroweb/projects/2/retros



Ergebnis in Postman bei Aufruf: http://localhost:8080/retroweb/projects/77/retros



Problem mit Lazy Loading bei Entity Retro

```
@RequestMapping(value = "/projects/{id}/retros", method = RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE)
public List<Retro> getAllRetros(@PathVariable long id, HttpServletRequest request, HttpServletResponse response) {
    response.setHeader("Cache-Control", "no-cache");
    List <Retro> newList = new ArrayList<>();
    newList = retroService.getByProjectId(id);
    return newList;
}
```

org.springframework.http.converter.HttpMessageConversionException:

Type definition error: [simple type, class
org.hibernate.proxy.pojo.javassist.JavassistLazyInitializer]; nested
exception is
com.fasterxml.jackson.databind.exc.InvalidDefinitionException: No
serializer found for class
org.hibernate.proxy.pojo.javassist.JavassistLazyInitializer and no
properties discovered to create BeanSerializer (to avoid exception,
disable SerializationFeature.FAIL_ON_EMPTY_BEANS) (through
reference chain: java.util.ArrayList[0]>de.htwg.retroweb.entities.Retro["project"]>de.htwg.retroweb.entities.Project \$\$ jvst5e 0["handler"])

Lazy Loading-Problem:

Wenn die Serialisierung nach JSON beim Return in der Methode startet, ist die Property project noch nicht geladen, es sei denn es ist im Cache von Hibernate, d.h. es wurde vorher schon mal abgefragt.

```
@ManyToOne(fetch = FetchType.LAZY, optional = false)
@JoinColumn(name = "project_id", nullable = false)
private Project project;
```

Lösung 1 für Lazy Loading bei Entity Retro

```
RestController-Methode
@RequestMapping(value = "/projects/{id}/retros", method = RequestMethod. GET, produces = MediaType.APPLICATION JSON VALUE)
 public List<Retro> getAllRetros(@PathVariable long id, HttpServletRequest request, HttpServletResponse response) {
   response.setHeader("Cache-Control", "no-cache");
   List <Retro> newList = new ArrayList<>();
                                                                 Lazy-Loading vermeiden bei REST
   newList = retroService.getByProjectId(id);
   return newList;
                                    Retro-Entity
                @ManyToOne(fetch = FetchType.EAGER, optional = false)
                @JoinColumn(name = "project id", nullable = false)
                private Project project;
```

Lösung 2 für Lazy Loading bei Entity Retro

```
@RequestMapping(value = "projects/{id}/retros", method = RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE)
public List<Retro> getRetrosFromProject(@PathVariable(value = "id") Long projectId, HttpServletResponse response) {
    Project project;
    try {
        project = projectService.getById(projectId);
        return retroService.findByProject(project);
    } catch (ResourceNotFoundException e) {
        response.setStatus(404);
        return new ArrayList<Retros>();
    }
}
Projekt muss vorher einmal geladen werden,
    so dass es im Cache von Hibernate ist.
```

```
@ManyToOne(fetch = FetchType.LAZY, optional = false)
@JoinColumn(name = "project_id", nullable = false)
private Project project;
```