

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.

```
package de.htwg.retroweb.controller.rest;
```

```
import java.util.ArrayList;  
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;  
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;  
import de.htwg.retroweb.service.ProjectsService;  
import de.htwg.retroweb.service.RetrosService;
```

Controller

Annotation, dass es sich um einen Rest-Controller handelt!!!

```
@RestController  
public class ExerciseRetrosRestController {
```

```
    @Autowired  
    private RetroService retroService;
```

```
    @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);  
            return retroService.findByProject(project);  
        } catch (ResourceNotFoundException e) {  
            response.setStatus(404);  
            return new ArrayList<Retros>();  
        }  
    }  
}
```

RequestMapping für die REST-Url

Es soll JSON zurückgeliefert werden.

Falls das Projekt nicht existiert, wird ein 404 zurückgegeben und eine leere Retro-Liste.

Zuerst wird das Projekt geholt, dann mit dem Objekt project werden die Retros für das Projekt geholt und zurückgegeben. Die Konvertierung nach JSON geht automatisch.

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

The screenshot shows a Postman interface with a GET request to `http://localhost:8080/retroweb/projects/2/retros`. The status is `200 OK`. The response is a JSON array of two retros, labeled `retro2` and `retro3`, which are grouped by a bracket labeled `Array von Retros`. A text box explains that the project with `id=2` has two retros.

```
[{"created": "2018-08-10T13:38:58.000+0000", "updated": "2018-08-10T13:38:58.000+0000", "id": 2, "name": "retro2", "active": false, "scheduled": "2018-08-17T13:30:00.000+0000", "project": {"created": "2018-08-10T13:38:58.000+0000", "updated": "2018-08-10T13:38:58.000+0000", "id": 2, "name": "project2", "active": true}}, {"created": "2018-08-10T13:38:58.000+0000", "updated": "2018-08-10T13:38:58.000+0000", "id": 3, "name": "retro3", "active": true, "scheduled": "2018-09-05T13:00:00.000+0000", "project": {"created": "2018-08-10T13:38:58.000+0000", "updated": "2018-08-10T13:38:58.000+0000", "id": 2, "name": "project2", "active": true}}]
```

retro2

retro3

Array von Retros

Das Projekt mit der id=2 hat zwei Retros.

Response-Code 200

Status: 200 OK

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

The screenshot shows the Postman interface. At the top, a GET request is defined for the URL `http://localhost:8080/retroweb/projects/77/retros`. The 'Send' button is highlighted. On the right, the status is indicated as 'Status: 404 Not Found'. Below the request bar, the response is displayed in JSON format, showing an empty array `[]`. Two blue arrows point from text boxes to the response: one points to the `[]` and the other points to the 'Status: 404 Not Found' text.

GET `http://localhost:8080/retroweb/projects/77/retros` Params Send Save

Status: 404 Not Found

Pretty Raw Preview JSON `[]`

1 `[]`

Leerer Array, das Projekt mit der id=77 wurde nicht gefunden, also keine Retros.

Response-Code 404

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<>();  
    newList = retroService.getByProjectId(id);  
    return newList;  
}
```

Lazy-Loading vermeiden bei REST

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;
```