Arbeitsblatt 5 – Dispatching in SpringMVC

Ziele

Sie kennen das **Dispatching von SpringMVC** und können dieses korrekt einsetzen.

Ausgangslage

Sie haben das Arbeitsblatt 3 korrekt gelöst – oder haben die Lösungen vom AD unter "Lektion02/AB3_UB2-solutions/flashcard-basic.zip" in ihre IDE importiert.

Aufgabe 1: Domain erstellen

Kopieren sie die Klasse "Questionnaire" in das Package "ch.fhnw.webfr.flashcard.domain".

Aufgabe 2: Persistence erstellen

Kopieren sie die Klasse "QuestionnaireRepository" in das Package "ch.fhnw.webfr.flashcard.persistence" und passen sie die Klasse "QuestionnaireRepository" an, wie in Listing 1 dargestellt.

```
@Component                                                              #1
public class QuestionnaireRepository {
  private static final Log logger =
           LogFactory.getLog(QuestionnaireRepository.class);
  private List<Questionnaire> questionnaires =
           new ArrayList<Questionnaire>();

  public Long save(Questionnaire q) {
    Long id = new Long(questionnaires.size());
    q.setId(id);
    questionnaires.add(q);
    return id;
  }

  public List<Questionnaire> findAll() {
    return questionnaires;
  }

  public Questionnaire findById(Long id) {
    return questionnaires.get(id.intValue());
  }

  public void clear() {
    questionnaires = new ArrayList<Questionnaire>();
  }

  @PostConstruct                                                        #2
  public void initRepoWithTestData() {
    save(new Questionnaire("Test Questionnaire 1",
         "Lorem ipsum dolor sit amet..."));
    save(new Questionnaire("Test Questionnaire 2",
         "Lorem ipsum dolor sit amet..."));
    save(new Questionnaire("i18n Test fünf",
         "Lorem ipsum dolor sit amet..."));
    logger.debug("Questionnaires initialized successfully");            #3
  }
}
```

Listing 1: "QuestionnaireRepository" als Spring Bean (→ @Component)

Hinweis:
#1: Mit dieser Annotation wird die Klasse vom Spring Container als Spring Bean erkannt. Wichtig!
#2: Die Methode bei dieser Annotation wird der Spring Container nach der Initialisierung des Spring Bean ausführen. In diesem Falle werden ein paar Questionnaire-Entitäten angelegt.
#3: Diesen Log-Output sollten sie beim Hochfahren der Applikation in der Console sehen.

Starten sie die Applikation. Kontrollieren sie den Log-Output. Es sollte kein Fehler auftauchen.

Aufgabe 3: "BasicServlet" zu Page Controller umwandeln
Erstellen sie die Klasse "ch.fhnw.webfr.flashcard.web.QuestionnareController" mit den beiden Methoden:

```
public void findAll(HttpServletResponse response, HttpServletRequest request)
              throws IOException {
}

public void findById(Long id, HttpServletResponse response,
              HttpServletRequest request) throws IOException {
}
```

Nun kopieren sie aus dem "BasicServlet" den Inhalt der entsprechenden Methoden in diese neuen Methoden:
"findAll()" - siehe "handleQuestionnairesRequest()"
"findById()" – siehe "handleQuestionnaireIdRequest()"

Sie brauchen nun im Controller eine Referenz auf das "QuestionnaireRepository". Diese können sie mittels Dependency Injection über die @Autowired Annotation erreichen. Kopieren sie folgende Zeile in den Controller.

```
@Autowired
private QuestionnaireRepository questionnaireRepository;
```

Ergänzen sie im "QuestionnaireController" alle fehlenden Annotationen, so dass die Mapping für die Handler-Methoden korrekt aufgebaut werden kann.

Hinweis: Beachten sie die Folie mit den Bemerkungen zu @RequestMapping und überprüfen sie die URL, die sie in ihrem Code erstellen mit den entsprechenden Vorgaben aus ihren RequestMapping's.

Aufgabe 4: Webapplikation testen
Testen sie die Antwort auf folgende URL Requests:

http://localhost:8080/flashcard-mvc/questionnaires
http://localhost:8080/flashcard-mvc/questionnaires/1

Sie müssen den gleichen Output erhalten wie in Arbeitsblatt 3.

Hinweis: Damit die Webapp unter dem Kontext "flashcard-mvc" sichtbar ist, dürfen sie im File "application.properties" den Eintrag gemäss Listing 1, Arbeitsblatt 4 nicht vergessen.