Software Engineering 2

Lösungen der Übung

Dr. F.-K. Koschnick, Sybit GmbH

Aufgabe (Story): JUnit-Test für Controller der Projektseite

Als Entwickler möchte ich einen JUnit-Test für meinen neugebauten ProjectController haben, damit ich diesen bei Weiterentwicklungen ändern kann und der JUnit-Test mir mehr Sicherheit gibt, dass der ProjectController weiterhin funktioniert.
Verifiziere, dass
es eine JUnit-Testklasse für den ProjectController gibt, die das Mockito-Testframework nutzt
diese Testklasse 2 Testmethoden hat: testShowProject() und testShowProjectWithoutLogin()
testShowProject() Folgendes prüft:
-- Anzahl der Attribute des Models (3), das sind: userName, project, retros
-- Response Code
-- Name des Views
-- forwardedUrl
testShowProjectWithoutLogin() Folgendes prüft:
-- Anzahl der Attribute des Models (0)
-- Response Code 302
-- view (redirect:/login)
-- redirectedUrl
Tipp:
Den SessionService mocken und die MockHttpSession nutzen, um die Session zu simulieren.
Schaut bitte in den JUnit-Test *HomeControllerTest.java*. Dort ist ähnliches für den HomeController gemacht. Bitte aber nicht einfach abschreiben sondern auch verstehen.

Testklasse und „Mocken“

```
@RunWith(SpringRunner.class)
@WebMvcTest(ProjectController.class)
@AutoConfigureMockMvc
public class ProjectControllerTest {
@Autowired
private MockMvc mvc;
@MockBean
private ProjectService projectService;
@MockBean
private RetroService retroService;
@MockBean
private SessionService sessionService;
@Mock
private Project project;
@Mock
private List<Retro> retros;

private MockHttpSession mockSession;
```

Konfiguration, dass das Spring gestartet wird

Annotationen, dass sich Test auf ProjectController bezieht und dass der Request, der View und das Model simuliert (gemockt) werden. Die Simulation mit dem Controller steht dann im Objekt mvc der Klasse MockMvc zur Verfügung.

Alle Services werden mit Mockbeans (Simulationen) „weggemockt“. Die Mockbeans implementieren automatisch die Interfaces der Services. Damit haben wir als Implementationen der Services dem Controller die MockBeans untergeschoben.

Mocks auch für Entities

Auch die Session wird durch eine Mock-Session ersetzt, da ja der WebServer für diesen Test nicht zur Verfügung steht.

Initialisierung mit Junit-Annotation @Before

```
@Before
public void initializeData(){
        mockSession = new MockHttpSession();
}
```

Die MockSession muss initialisiert werden.

Die Methode initializeData() mit der Annotation @Before wird vor jedem Test, also vor der Ausführung jeder Testmethode, ausgeführt.

Die Testmethode testShowProject()

(für die Controller-Methode showProject(...))

```
@Test
public void testShowProject() throws Exception {
when(sessionService.isLoggedIn(mockSession)).thenReturn(true);
when(sessionService.getUserName(mockSession)).thenReturn("test");

when(projectService.getById((long) 1)).thenReturn(project);

when(retroService.findByProject(project)).thenReturn(retros);

mvc.perform(MockMvcRequestBuilders.get("/project").param("id",
"1").session(mockSession))
      .andExpect(model().size(3))
      .andExpect(model().attribute("userName", "test"))
      .andExpect(model().attribute("project", project))
      .andExpect(model().attribute("retros", retros))
.andExpect(status().isOk())
      .andExpect(view().name("project"))
      .andExpect(forwardedUrl("/WEB-INF/jsp/project.jsp"));
}
```

Festlegen, wie die gemockten, leeren Services reagieren sollen.
Z.B.: wenn die isLoggedIn-Methode des gemockten Session-Service mit der Mock-Session als Argument aufgerufen wird, soll true zurückgegeben werden. Es wird dann simuliert, dass die Session (hier Mock-Session) gültig ist und der User eingeloggt ist.

mvc.perform bedeutet, den User-Request zu simulieren und die Controller-Methode showProject zu starten, die über Requestmapping mit der rel. URL /project verbunden ist.

Mit andExpect werden dann verschiedene Prüfungen am Model oder Status, View, etc. durchgeführt. Sind diese Prüfungen erfolgreich, so arbeitet der Controller wie gewünscht.

Die Testmethode testShowProjectWithoutLogin()

(für die Controller-Methode showProject(…), wenn die Session ungültig ist)

```
@Test
public void testShowProjectWithoutLogin() throws Exception {
when(sessionService.isLoggedIn(mockSession)).thenReturn(false);

mvc.perform(MockMvcRequestBuilders.get("/project").param("id",
"1").session(mockSession))
.andExpect(model().size(0))
.andExpect(status().is(302))
.andExpect(view().name("redirect:/login"))
.andExpect(redirectedUrl("/login"));
}
```

Hier wird mit der Mock-Session simuliert, dass die Session ungültig ist, der User also nicht eingeloggt ist. Daher wird jetzt false zurückgeliefert.

mvc.perform(…) und entsprechende Prüfungen mit andExcept() wie gehabt. Diesmal muss auf die Login-Seite mit einem Redirect zurückverwiesen werden.

(Anhang) Package und Imports der Testklasse

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

import static org.mockito.Mockito.when;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.forwardedUrl;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.model;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.redirectedUrl;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.view;

import java.util.ArrayList;
import java.util.List;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.mock.web.MockHttpSession;
import org.springframework.test.context.junit4.SpringRunner;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;

import de.htwg.retroweb.entities.Project;
import de.htwg.retroweb.entities.Retro;
import de.htwg.retroweb.service.ProjectService;
import de.htwg.retroweb.service.RetroService;
import de.htwg.retroweb.service.SessionService;
```