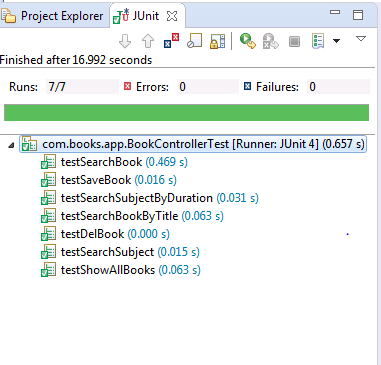
**SPRING TEST ASSIGNMENT 2**

**Results**



**BookControllerTest.java**

**package** com.books.app;

**import** **static** java.util.Collections.*singletonList*;

**import** **static** org.hamcrest.CoreMatchers.*hasItem*;

**import** **static** org.hamcrest.CoreMatchers.~~is~~;

**import** **static** org.hamcrest.Matchers.*allOf*;

**import** **static** org.hamcrest.Matchers.*hasProperty*;

**import** **static** org.mockito.BDDMockito.*given*;

**import** **static** org.mockito.Mockito.*times*;

**import** **static** org.mockito.Mockito.*verify*;

**import** **static** org.mockito.Mockito.*verifyNoMoreInteractions*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*get*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*post*;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.*model*;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.*status*;

**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.WebMvcTest;

**import** org.springframework.boot.test.mock.mockito.MockBean;

**import** org.springframework.test.context.junit4.SpringRunner;

**import** org.springframework.test.web.servlet.MockMvc;

**import** org.springframework.test.web.servlet.result.MockMvcResultMatchers;

**import** org.springframework.test.web.servlet.setup.MockMvcBuilders;

**import** org.springframework.web.context.WebApplicationContext;

**import** com.books.app.controller.BookController;

**import** com.books.app.model.Book;

**import** com.books.app.model.Subject;

**import** com.books.app.service.IBookAppService;

@RunWith(SpringRunner.**class**)

@WebMvcTest(controllers = BookController.**class**)

**public** **class** BookControllerTest {

**private** MockMvc mockMvc;

@Autowired

**private** WebApplicationContext webApplicationContext;

@MockBean

**private** IBookAppService bookAppService;

@Before

**public** **void** setUp() {

mockMvc = MockMvcBuilders.*webAppContextSetup*(webApplicationContext).build();

}

@SuppressWarnings("unchecked")

@Test

**public** **void** testShowAllBooks() **throws** Exception {

Book book = **new** Book();

book.setTitle("Species");

List<Book> allBooks = *singletonList*(book);

*given*(bookAppService.view()).willReturn(allBooks);

mockMvc.perform(*get*("/view")).andExpect(*status*().isOk()).andExpect(MockMvcResultMatchers.*model*()

.attribute("booklist", *hasItem*(*allOf*(*hasProperty*("title", *is*("Species"))))));

*verify*(bookAppService, *times*(1)).view();

*verifyNoMoreInteractions*(bookAppService);

}

@Test

**public** **void** testSearchBook() **throws** Exception {

Subject sub1 = **new** Subject();

sub1.setSubtitle("Subject 1");

Book book1 = **new** Book();

book1.setBookId(1);

book1.setTitle("Book 1");

book1.setPrice(10.00);

book1.setVolume(1);

book1.setSubject(sub1);

*given*(bookAppService.search(1)).willReturn(book1);

mockMvc.perform(*get*("/searchBook/1")).andExpect(*status*().isOk())

.andExpect(*model*().attribute("stitle", "Subject 1")).andExpect(*model*().attribute("bookId", 1))

.andExpect(*model*().attribute("title", "Book 1")).andExpect(*model*().attribute("price", 10.00))

.andExpect(*model*().attribute("volume", 1));

*verify*(bookAppService, *times*(1)).search(1);

*verifyNoMoreInteractions*(bookAppService);

}

@Test

**public** **void** testSearchBookByTitle() **throws** Exception {

Subject sub1 = **new** Subject();

sub1.setSubtitle("Subject 1");

Book book1 = **new** Book();

book1.setBookId(1);

book1.setTitle("Book 1");

book1.setPrice(10.00);

book1.setVolume(1);

book1.setSubject(sub1);

*given*(bookAppService.searchByTitle("Book 1")).willReturn(book1);

mockMvc.perform(*get*("/searchBookByTitle").param("title", "Book 1")).andExpect(*status*().isOk())

.andExpect(*model*().attribute("stitle", "Subject 1")).andExpect(*model*().attribute("bookId", 1))

.andExpect(*model*().attribute("title", "Book 1")).andExpect(*model*().attribute("price", 10.00))

.andExpect(*model*().attribute("volume", 1));

*verify*(bookAppService, *times*(1)).searchByTitle("Book 1");

*verifyNoMoreInteractions*(bookAppService);

}

@Test

**public** **void** testSearchSubjectByDuration() **throws** Exception {

List<Subject> subjList = **new** ArrayList<Subject>();

Subject sub1 = **new** Subject();

sub1.setSubtitle("Subject 1");

sub1.setSubjectId(1);

sub1.setDurationInHours(20);

Subject sub2 = **new** Subject();

sub2.setSubtitle("Subject 2");

sub2.setSubjectId(2);

sub2.setDurationInHours(20);

subjList.add(sub1);

subjList.add(sub2);

*given*(bookAppService.searchSubjectByDuration(20)).willReturn(subjList);

mockMvc.perform(*get*("/searchSubjectByDuration").param("durationInHours", String.*valueOf*(20)))

.andExpect(*status*().isOk()).andExpect(*model*().attribute("subjList", subjList));

*verify*(bookAppService, *times*(1)).searchSubjectByDuration(20);

*verifyNoMoreInteractions*(bookAppService);

}

@Test

**public** **void** testSearchSubject() **throws** Exception {

Subject sub1 = **new** Subject();

sub1.setSubtitle("Subject 1");

sub1.setSubjectId(1);

sub1.setDurationInHours(20);

*given*(bookAppService.searchSubject("Subject 1")).willReturn(sub1);

mockMvc.perform(*get*("/searchSubject").param("subtitle", "Subject 1")).andExpect(*status*().isOk())

.andExpect(*model*().attribute("subtitle", "Subject 1"));

*verify*(bookAppService, *times*(1)).searchSubject("Subject 1");

*verifyNoMoreInteractions*(bookAppService);

}

@Test

**public** **void** testSaveBook() **throws** Exception {

Subject sub1 = **new** Subject();

sub1.setSubtitle("Subject 1");

Book book1 = **new** Book();

book1.setBookId(1);

book1.setTitle("Book 1");

book1.setPrice(10.00);

book1.setVolume(1);

book1.setSubject(sub1);

*given*(bookAppService.add(book1, sub1.getSubtitle())).willReturn(book1.getTitle());

mockMvc.perform(*post*("/saveBook").sessionAttr("book", book1)).andExpect(*status*().isOk())

.andExpect(*model*().attribute("title", "Book 1"));

*verify*(bookAppService, *times*(1)).add(book1, sub1.getSubtitle());

*verifyNoMoreInteractions*(bookAppService);

}

@Test

**public** **void** testDelBook() **throws** Exception {

Subject sub1 = **new** Subject();

sub1.setSubtitle("Subject 1");

Book book1 = **new** Book();

book1.setBookId(1);

book1.setTitle("Book 1");

book1.setPrice(10.00);

book1.setVolume(1);

book1.setSubject(sub1);

bookAppService.delete(1);

*verify*(bookAppService, *times*(1)).delete(book1.getBookId());

*verifyNoMoreInteractions*(bookAppService);

}

}