**SPRING TEST ASSIGNMENT 3**

**BookRepositoryTest.java**

**package** com.books.app;

**import** **static** org.assertj.core.api.Assertions.*assertThat*;

**import** java.time.LocalDate;

**import** java.util.List;

**import** java.util.Optional;

**import** org.junit.Test;

**import** org.junit.runner.RunWith;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;

**import** org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase.Replace;

**import** org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;

**import** org.springframework.boot.test.autoconfigure.orm.jpa.TestEntityManager;

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

**import** com.books.app.dao.IBookRepository;

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

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

@RunWith(SpringRunner.**class**)

@DataJpaTest

@AutoConfigureTestDatabase(replace = Replace.***NONE***)

**public** **class** BookRepositoryTest {

@Autowired

**private** TestEntityManager entityManager;

@Autowired

**private** IBookRepository bookRepository;

@Test

**public** **void** whenFindById\_thenReturnBook() {

// given

Subject sub = **new** Subject();

sub.setSubtitle("Subject 1");

entityManager.persist(sub);

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

book.setBookId(1);

book.setTitle("Book 1");

book.setPrice(10.00);

book.setVolume(1);

book.setPublistDt(LocalDate.*now*());

book.setSubject(sub);

entityManager.persist(book);

entityManager.flush();

// when

Optional<Book> optbook = bookRepository.findById(book.getBookId());

Book sbook = optbook.get();

// then

*assertThat*(sbook.getTitle())

.isEqualTo(book.getTitle());

}

@Test

**public** **void** saveBook() {

// given

Subject sub = **new** Subject();

sub.setSubtitle("Subject 1");

entityManager.persist(sub);

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

book.setBookId(1);

book.setTitle("Book 1");

book.setPrice(10.00);

book.setVolume(1);

book.setPublistDt(LocalDate.*now*());

book.setSubject(sub);

entityManager.persist(book);

entityManager.flush();

// when

Book sbook = bookRepository.save(book);

// then

*assertThat*(sbook.getTitle())

.isEqualTo(book.getTitle());

}

@Test

**public** **void** deleteBook\_ById() {

// given

Subject sub = **new** Subject();

sub.setSubtitle("Subject 1");

entityManager.persist(sub);

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

book.setBookId(1);

book.setTitle("Book 1");

book.setPrice(10.00);

book.setVolume(1);

book.setPublistDt(LocalDate.*now*());

book.setSubject(sub);

entityManager.persist(book);

entityManager.flush();

// when

bookRepository.deleteById(book.getBookId());

// then

*assertThat*(bookRepository.findById(book.getBookId())).isEmpty();

}

@Test

**public** **void** whenFindAll\_thenReturnBookList() {

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

sub1.setSubtitle("Subject 1");

entityManager.persist(sub1);

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

book.setBookId(1);

book.setTitle("Book 1");

book.setPrice(10.00);

book.setVolume(1);

book.setPublistDt(LocalDate.*now*());

book.setSubject(sub1);

entityManager.persist(book);

entityManager.flush();

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

sub2.setSubtitle("Subject 2");

entityManager.persist(sub2);

Book book2 = **new** Book();

book2.setBookId(2);

book2.setTitle("Book 2");

book2.setPrice(20.00);

book2.setVolume(2);

book2.setPublistDt(LocalDate.*now*());

book2.setSubject(sub2);

entityManager.persist(book2);

entityManager.flush();

List<Book> bookList = bookRepository.findAll();

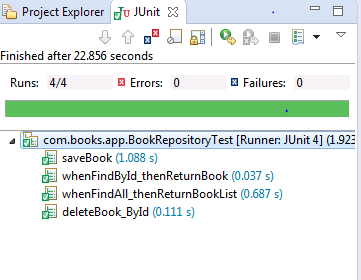
*assertThat*(bookList.get(0).getTitle())

.isEqualTo("Book 1");

}

}

**Results**

****

**SubjectRepositoryTest.java**

**package** com.books.app;

**import** **static** org.assertj.core.api.Assertions.*assertThat*;

**import** java.util.HashSet;

**import** java.util.List;

**import** java.util.Optional;

**import** java.util.Set;

**import** org.junit.Test;

**import** org.junit.runner.RunWith;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;

**import** org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase.Replace;

**import** org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;

**import** org.springframework.boot.test.autoconfigure.orm.jpa.TestEntityManager;

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

**import** com.books.app.dao.ISubjectRepository;

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

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

@RunWith(SpringRunner.**class**)

@DataJpaTest

@AutoConfigureTestDatabase(replace = Replace.***NONE***)

**public** **class** SubjectRepositoryTest {

@Autowired

**private** TestEntityManager entityManager;

@Autowired

**private** ISubjectRepository subjectRepository;

@Test

**public** **void** whenFindById\_thenReturnSubject() {

// given

Subject sub = **new** Subject();

sub.setSubtitle("Subject 1");

sub.setSubjectId(1);

sub.setDurationInHours(20);

Set<Book> references = **new** HashSet<Book>();

sub.setReferences(references);

entityManager.persist(sub);

entityManager.flush();

// when

Optional<Subject> optSubject = subjectRepository.findById(sub.getSubtitle());

Subject subj = optSubject.get();

// then

*assertThat*(sub.getSubtitle())

.isEqualTo(subj.getSubtitle());

}

@Test

**public** **void** saveSubject() {

// given

Subject sub = **new** Subject();

sub.setSubtitle("Subject 1");

sub.setSubjectId(1);

sub.setDurationInHours(20);

Set<Book> references = **new** HashSet<Book>();

sub.setReferences(references);

entityManager.persist(sub);

entityManager.flush();

// when

Subject subj = subjectRepository.save(sub);

// then

*assertThat*(sub.getSubtitle())

.isEqualTo(subj.getSubtitle());

}

@Test

**public** **void** deleteSubject\_ById() {

// given

Subject sub = **new** Subject();

sub.setSubtitle("Subject 1");

sub.setSubjectId(1);

sub.setDurationInHours(20);

Set<Book> references = **new** HashSet<Book>();

sub.setReferences(references);

entityManager.persist(sub);

entityManager.flush();

// when

subjectRepository.deleteById(sub.getSubtitle());

// then

*assertThat*(subjectRepository.findById(sub.getSubtitle())).isEmpty();

}

@Test

**public** **void** whenFindAll\_thenReturnSubjectList() {

// given

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

sub1.setSubtitle("Subject 1");

sub1.setSubjectId(1);

sub1.setDurationInHours(10);

Set<Book> references1 = **new** HashSet<Book>();

sub1.setReferences(references1);

entityManager.persist(sub1);

entityManager.flush();

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

sub2.setSubtitle("Subject 2");

sub2.setSubjectId(2);

sub2.setDurationInHours(20);

Set<Book> references2 = **new** HashSet<Book>();

sub2.setReferences(references2);

entityManager.persist(sub2);

entityManager.flush();

// when

List<Subject> subjList = **null**;

subjList = subjectRepository.findAll();

**for** (Subject subject : subjList) {

System.***out***.println(subject.getSubtitle());

}

// then

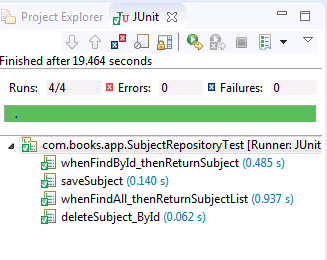
*assertThat*(subjList.stream().filter(sub -> "Subject 1".equalsIgnoreCase(sub.getSubtitle())).findFirst().isPresent())

.isEqualTo(**true**);

}

}

**Results**

****