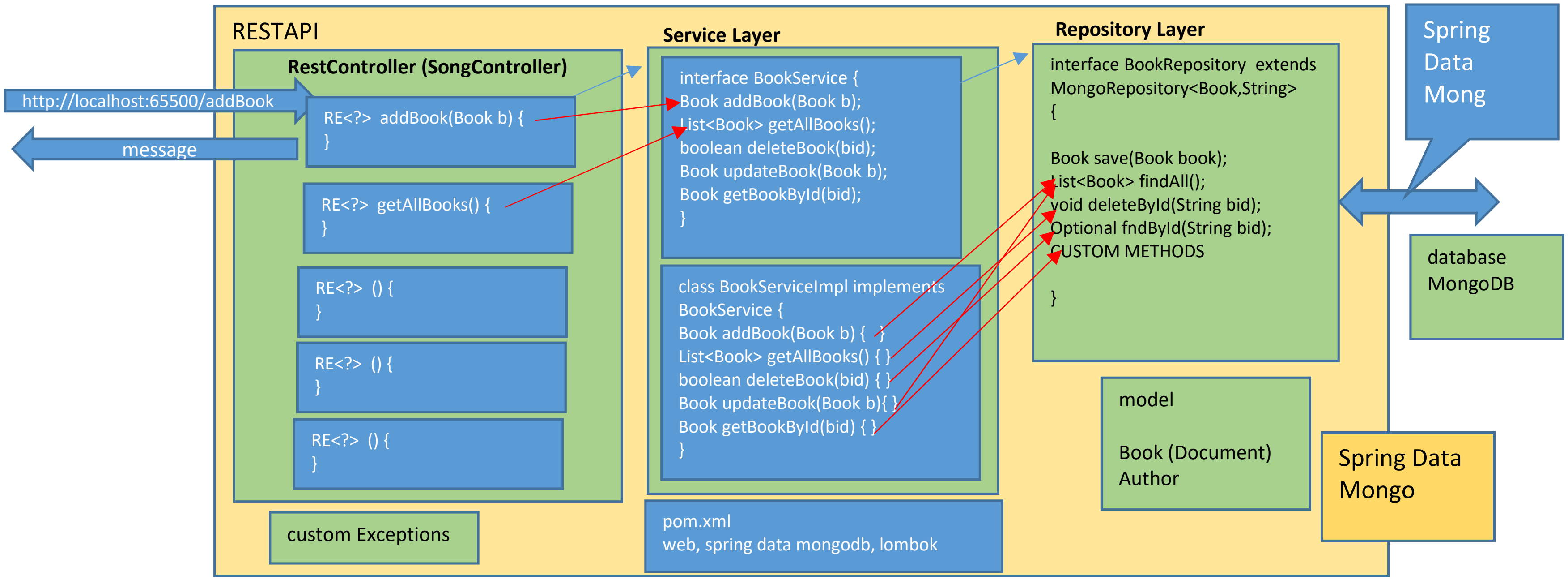


RESTAPI

http://localhost:65500/addBook

message

RestController (SongController)

```
RE<?>  addBook(Book b) {
}

RE<?>  getAllBooks() {
}

RE<?>  () {
}

RE<?>  () {
}

RE<?>  () {
}
```

custom Exceptions

Service Layer

```
interface BookService {
Book addBook(Book b);
List<Book> getAllBooks();
boolean deleteBook(bid);
Book updateBook(Book b);
Book getBookById(bid);
}
```

```
class BookServiceImpl implements
BookService {
Book addBook(Book b) {  }
List<Book> getAllBooks() { }
boolean deleteBook(bid) { }
Book updateBook(Book b){ }
Book getBookById(bid) { }
}
```

pom.xml
web, spring data mongodb, lombok

Repository Layer

```
interface BookRepository  extends
MongoRepository<Book,String>
{

Book save(Book book);
List<Book> findAll();
void deleteById(String bid);
Optional fndById(String bid);
CUSTOM METHODS

}
```

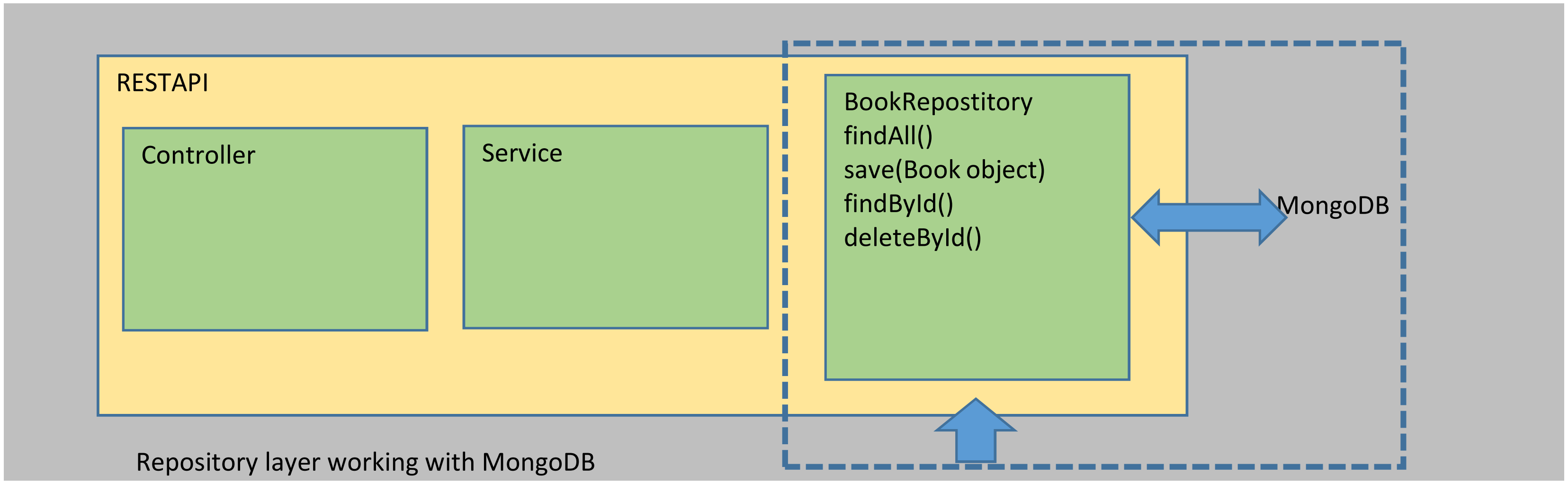
model

Book (Document)
Author

Spring Data Mongo

Spring
Data
Mong

database
MongoDB



RESTAPI

Controller

Service

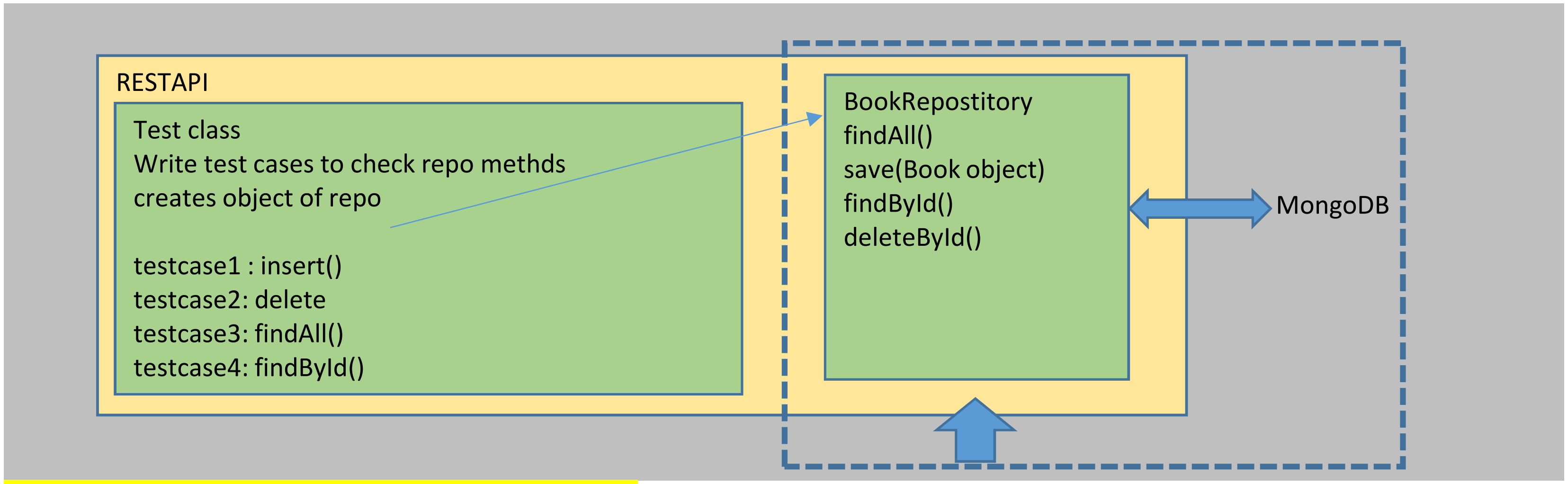
BookRepostitory
findAll()
save(Book object)
findById()
deleteById()

MongoDB

Repository layer working with MongoDB

Dependency is MongoDB
Dependent is Repository

Dependency is not under programmer control, so mocked env can not be created for DB



RESTAPI

Test class
Write test cases to check repo methds
creates object of repo

testcase1 : insert()
testcase2: delete
testcase3: findAll()
testcase4: findById()

BookRepostitory
findAll()
save(Book object)
findById()
deleteById()

MongoDB

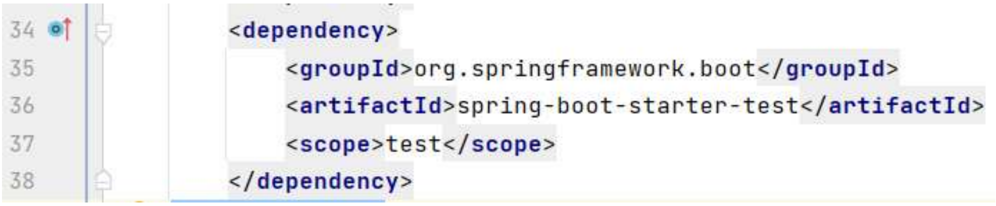
Steps to test repository layer

Step 1 Make sure springboot application is ready with all layers and with all functionalities

copy sprint3 demo (mongodb-with crud operations)

make sure mongodb is running

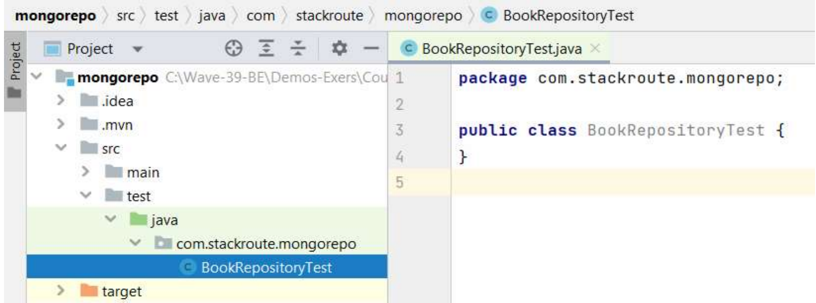
make sure testing dependency added in pom (default)



```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
```

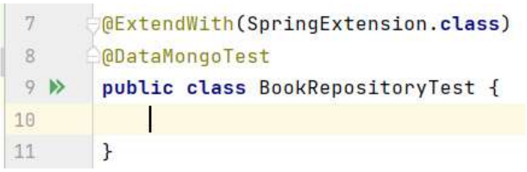
Step 2 Create testing class under testing folder

Note: delete default created testing file


mongorepo › src › test › java › com › stackroute › mongorepo › BookRepositoryTest
Project
mongorepo C:\Wave-39-BE\Demos-Exers\Cou
.idea
.mvn
src
main
test
java
com.stackroute.mongorepo
BookRepositoryTest
target
BookRepositoryTest.java

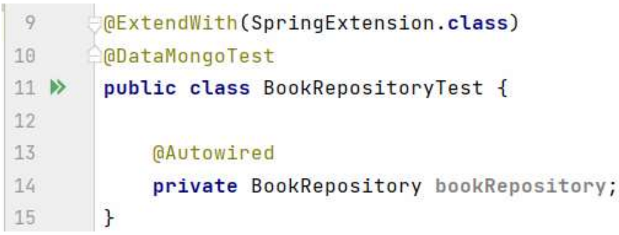
```
package com.stackroute.mongorepo;

public class BookRepositoryTest {
}
```

Step 3 Add required annotations on class



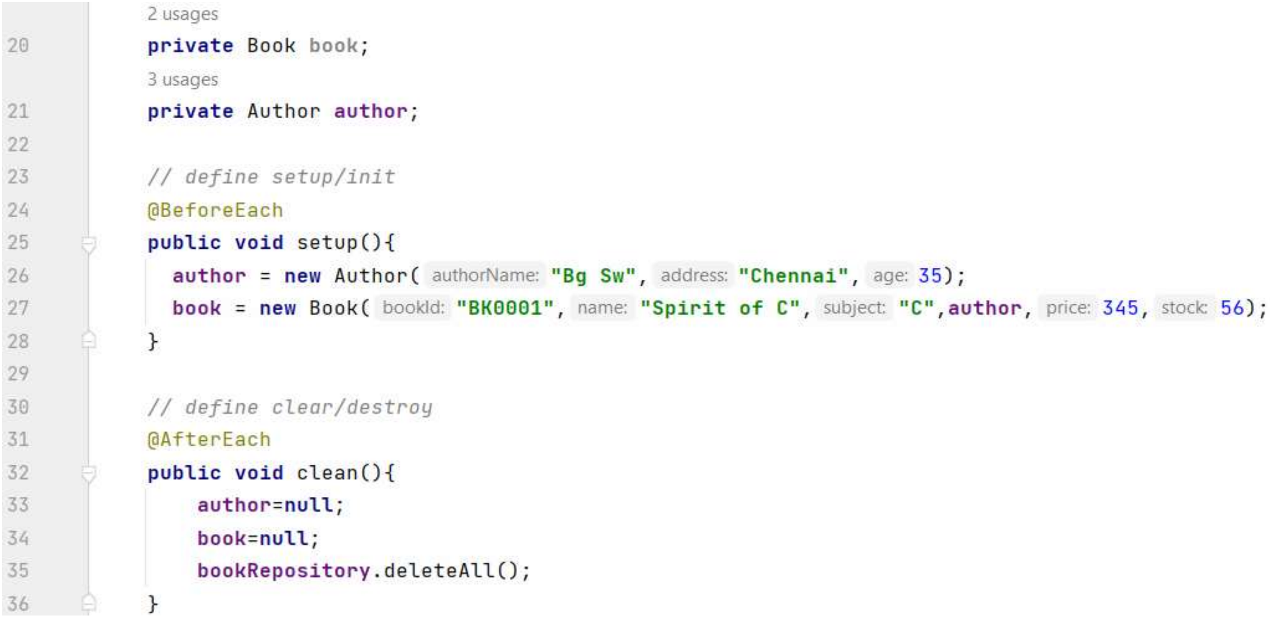
```
@ExtendWith(SpringExtension.class)
@DataMongoTest
public class BookRepositoryTest {
    
}
```

Step 4 Inject repository object as dependency into test class



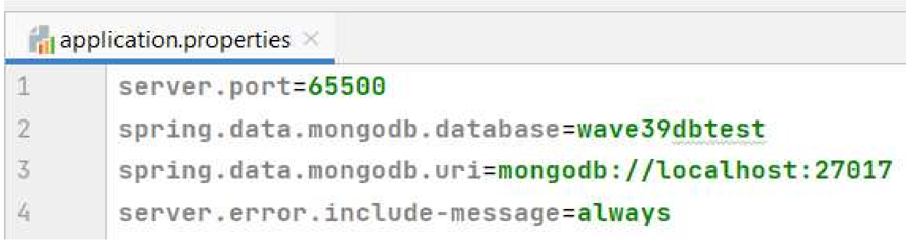
```
@ExtendWith(SpringExtension.class)
@DataMongoTest
public class BookRepositoryTest {

    @Autowired
    private BookRepository bookRepository;
}
```

Step 5 Define before each / after each



```
    2 usages
    private Book book;
    3 usages
    private Author author;

    // define setup/init
    @BeforeEach
    public void setup(){
      author = new Author( authorName: "Bg Sw", address: "Chennai", age: 35);
      book = new Book( bookId: "BK0001", name: "Spirit of C", subject: "C",author, price: 345, stock: 56);
    }

    // define clear/destroy
    @AfterEach
    public void clean(){
        author=null;
        book=null;
        bookRepository.deleteAll();
    }
```

Step 6 create new db for testing, mention the same in application.properties (optional)



```
server.port=65500
spring.data.mongodb.database=wave39dbtest
spring.data.mongodb.uri=mongodb://localhost:27017
server.error.include-message=always
```

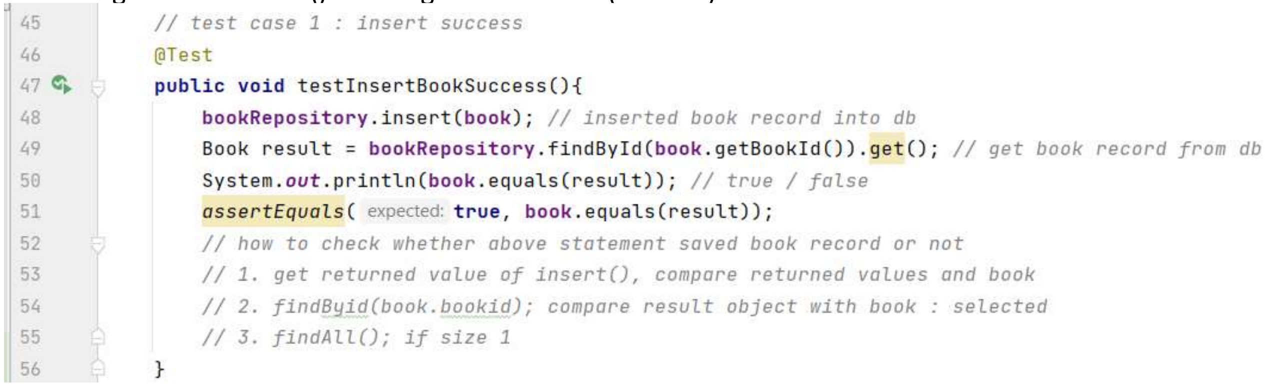
Step 7 write test cases

insert()

saves record and returns same object if id based not existing (SUCCESS)

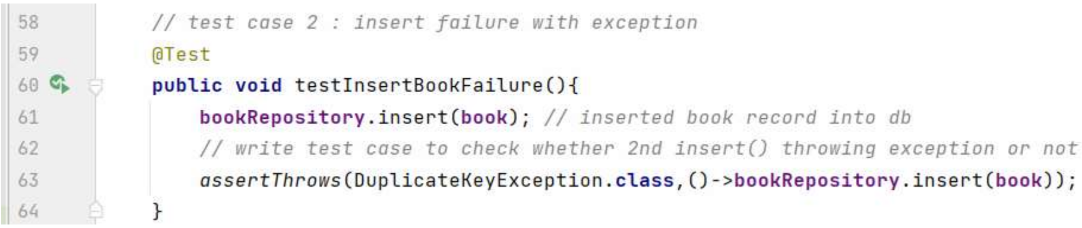
throws exception if ID based record is existing (Failure)

Test 1 checking whether insert() is saving record or not (Success)



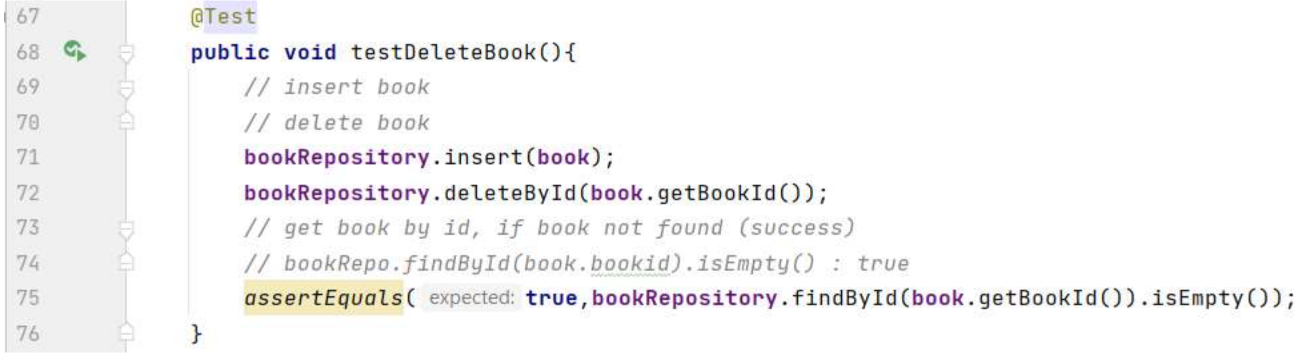
```
// test case 1 : insert success
@Test
public void testInsertBookSuccess(){
    bookRepository.insert(book); // inserted book record into db
    Book result = bookRepository.findById(book.getBookId()).get(); // get book record from db
    System.out.println(book.equals(result)); // true / false
    assertEquals( expected: true, book.equals(result));
    // how to check whether above statement saved book record or not
    // 1. get returned value of insert(), compare returned values and book
    // 2. findByid(book.bookid); compare result object with book : selected
    // 3. findAll(); if size 1
}
```

Test 2 checking whether insert() is saving record or not (Fail)



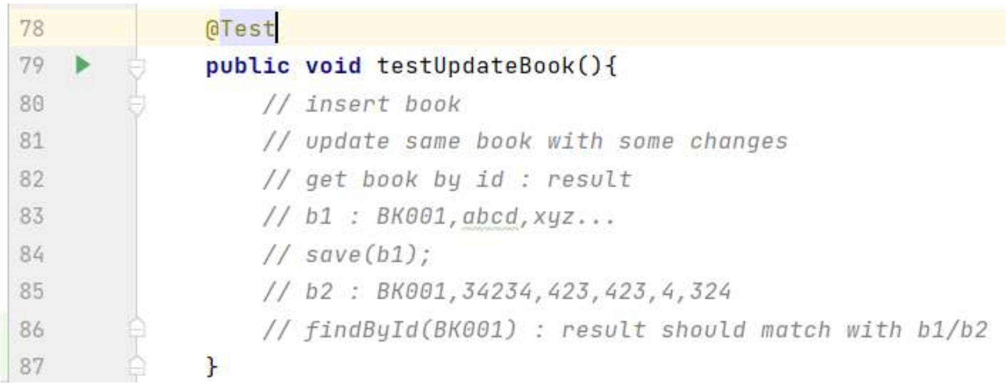
```
// test case 2 : insert failure with exception
@Test
public void testInsertBookFailure(){
    bookRepository.insert(book); // inserted book record into db
    // write test case to check whether 2nd insert() throwing exception or not
    assertThrows(DuplicateKeyException.class,()->bookRepository.insert(book));
}
```

Test 3 checking whether deleteById() working or not (success)



```
@Test
public void testDeleteBook(){
    // insert book
    // delete book
    bookRepository.insert(book);
    bookRepository.deleteById(book.getBookId());
    // get book by id, if book not found (success)
    // bookRepo.findById(book.bookid).isEmpty() : true
    assertEquals( expected: true,bookRepository.findById(book.getBookId()).isEmpty());
}
```

Test 4 checking updation



```
@Test
public void testUpdateBook(){
    // insert book
    // update same book with some changes
    // get book by id : result
    // b1 : BK001,abcd,xyz...
    // save(b1);
    // b2 : BK001,34234,423,423,4,324
    // findById(BK001) : result should match with b1/b2
}
```

Test 5 check findAll()