# Application.java

* package com.cognizant.fsd.springtestassignment1;
* import org.springframework.boot.SpringApplication;
* import org.springframework.boot.autoconfigure.SpringBootApplication;
* import org.springframework.boot.builder.SpringApplicationBuilder;
* import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;
* @SpringBootApplication
* public class Application extends SpringBootServletInitializer {
* @Override
* protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
* return application.sources(Application.class);
* }
* public static void main(String[] args) {
* SpringApplication.run(Application.class, args);
* }
* }

# SpringDataConfig.java

* package com.cognizant.fsd.springtestassignment1.config;
* import java.util.HashMap;
* import java.util.Properties;
* import javax.sql.DataSource;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.context.annotation.Bean;
* import org.springframework.context.annotation.Configuration;
* import org.springframework.context.annotation.PropertySource;
* import org.springframework.core.env.Environment;
* import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
* import org.springframework.jdbc.datasource.DriverManagerDataSource;
* import org.springframework.orm.jpa.JpaTransactionManager;
* import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;
* import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;
* import org.springframework.transaction.annotation.EnableTransactionManagement;
* @Configuration
* @EnableTransactionManagement
* @EnableJpaRepositories("com.cognizant.fsd.springtestassignment1.repository" )
* @PropertySource("classpath:mysql.properties")
* public class SpringDataConfig {
* private final String PROPERTY\_DRIVER = "jdbc.driverClassName";
* private final String PROPERTY\_URL = "jdbc.url";
* private final String PROPERTY\_USERNAME = "jdbc.user";
* private final String PROPERTY\_PASSWORD = "jdbc.pass";
* private final String PROPERTY\_SHOW\_SQL = "hibernate.show\_sql";
* private final String PROPERTY\_DIALECT = "hibernate.dialect";
* @Autowired
* Environment environment;
* /\* @Bean
* LocalContainerEntityManagerFactoryBean entityManagerFactory() {
* LocalContainerEntityManagerFactoryBean lfb = new LocalContainerEntityManagerFactoryBean();
* lfb.setDataSource(dataSource());
* //lfb.setPersistenceProviderClass(HibernatePersistence.class);
* lfb.setPackagesToScan("com.cognizant.fsd.springtestassignment1.model");
* lfb.setJpaProperties(hibernateProps());
* return lfb;
* }\*/
* @Bean
* public LocalContainerEntityManagerFactoryBean entityManagerFactory() {
* LocalContainerEntityManagerFactoryBean em
* = new LocalContainerEntityManagerFactoryBean();
* em.setDataSource(dataSource());
* em.setPackagesToScan(
* new String[] { "com.cognizant.fsd.springtestassignment1.model" });
* HibernateJpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter();
* em.setJpaVendorAdapter(vendorAdapter);
* HashMap<String, Object> properties = new HashMap<>();
* properties.put("hibernate.hbm2ddl.auto",
* environment.getProperty("hibernate.hbm2ddl.auto"));
* properties.put("hibernate.dialect",
* environment.getProperty("hibernate.dialect"));
* properties.put("hibernate.show\_sql",
* true);
* em.setJpaPropertyMap(properties);
* return em;
* }
* @Bean
* DataSource dataSource() {
* DriverManagerDataSource ds = new DriverManagerDataSource();
* ds.setUrl(environment.getProperty(PROPERTY\_URL));
* ds.setUsername(environment.getProperty(PROPERTY\_USERNAME));
* ds.setPassword(environment.getProperty(PROPERTY\_PASSWORD));
* ds.setDriverClassName(environment.getProperty(PROPERTY\_DRIVER));
* return ds;
* }
* Properties hibernateProps() {
* Properties properties = new Properties();
* properties.setProperty(PROPERTY\_DIALECT, environment.getProperty(PROPERTY\_DIALECT));
* properties.setProperty(PROPERTY\_SHOW\_SQL, environment.getProperty(PROPERTY\_SHOW\_SQL));
* return properties;
* }
* @Bean
* JpaTransactionManager transactionManager() {
* JpaTransactionManager transactionManager = new JpaTransactionManager();
* transactionManager.setEntityManagerFactory(entityManagerFactory().getObject());
* return transactionManager;
* }
* }

# FsdCollectionEditor.java

* package com.cognizant.fsd.springtestassignment1.controller;
* import java.util.Collection;
* import org.springframework.beans.propertyeditors.CustomCollectionEditor;
* import com.cognizant.fsd.springtestassignment1.model.Book;
* import com.cognizant.fsd.springtestassignment1.service.BookService;
* public class FsdCollectionEditor extends CustomCollectionEditor {
* private BookService bookService;
* public FsdCollectionEditor(Class<? extends Collection> collectionType, boolean nullAsEmptyCollection) {
* super(collectionType, nullAsEmptyCollection);
* }
* protected Object convertElement(Object element) {
* System.out.println("element : "+element);
* Book book = bookService.searchBook((Long)element);
* return book;
* }
* public BookService getBookService() {
* return bookService;
* }
* public void setBookService(BookService bookService) {
* this.bookService = bookService;
* }
* }

# HomeController.java

* package com.cognizant.fsd.springtestassignment1.controller;
* import java.util.Set;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.stereotype.Controller;
* import org.springframework.ui.Model;
* import org.springframework.web.bind.WebDataBinder;
* import org.springframework.web.bind.annotation.InitBinder;
* import org.springframework.web.bind.annotation.ModelAttribute;
* import org.springframework.web.bind.annotation.RequestMapping;
* import org.springframework.web.bind.annotation.RequestMethod;
* import org.springframework.web.bind.annotation.RequestParam;
* import com.cognizant.fsd.springtestassignment1.model.Book;
* import com.cognizant.fsd.springtestassignment1.model.Subject;
* import com.cognizant.fsd.springtestassignment1.service.BookService;
* import com.cognizant.fsd.springtestassignment1.service.SubjectService;
* @Controller
* public class HomeController {
* @Autowired
* private SubjectService subjectService;
* @Autowired
* private BookService bookService;
* @InitBinder
* public void dataBinding(WebDataBinder binder) {
* FsdCollectionEditor fsdCollectionEditor = new FsdCollectionEditor(Set.class, true);
* fsdCollectionEditor.setBookService(bookService);
* //binder.registerCustomEditor(Set.class, "references", fsdCollectionEditor);
* }
* @RequestMapping(path = { "/" }, method = RequestMethod.GET)
* public String index(Model model) {
* return "index";
* }
* @RequestMapping(value = "/addSubject", method = RequestMethod.GET)
* public String addSubject(Model model) {
* System.out.println("add Subject");
* Subject subject = new Subject();
* model.addAttribute("subject", subject);
* model.addAttribute("BOOK\_LIST", bookService.fetchAllBook());
* return "add-subject";
* }
* @RequestMapping(value = "/saveSubject", method = RequestMethod.POST)
* public String saveSubject(@ModelAttribute final Subject subject, Model model) {
* System.out.println("save Subject ");
* subject.getBookList().stream()
* .filter(bookId->bookId>0)
* .forEach(bookId->{
* subject.getReferences().add(bookService.searchBook(bookId));
* });
* model.addAttribute("ENTITY\_NAME", "Subject");
* subjectService.addSubject(subject);
* model.addAttribute("ENTITY\_ID", subject.getSubjectId());
* return "success";
* }
* @RequestMapping(value = "/addBook", method = RequestMethod.GET)
* public String addBook(Model model) {
* System.out.println("add Book");
* model.addAttribute("book", new Book());
* return "add-book";
* }
* @RequestMapping(value = "/saveBook", method = RequestMethod.POST)
* public String saveBook(@ModelAttribute Book book, Model model) {
* System.out.println("save book " + book.getPublishDate());
* model.addAttribute("ENTITY\_NAME", "Book");
* book = bookService.addBook(book);
* model.addAttribute("ENTITY\_ID", book.getBookId());
* return "success";
* }
* @RequestMapping(value = "/deleteBook", method = RequestMethod.GET)
* public String deleteBook(Model model) {
* System.out.println("delete Book");
* return "delete-book";
* }
* @RequestMapping(value = "/performDeleteBook", method = RequestMethod.GET)
* public String performDeleteBook(@RequestParam("bookId") String bookId, Model model) {
* System.out.println("delete Book " + bookId);
* boolean isSuccess = bookService.deleteBook(Long.parseLong(bookId));
* model.addAttribute("ENTITY\_NAME", "Book");
* model.addAttribute("ENTITY\_ID", bookId);
* model.addAttribute("isSuccess", isSuccess);
* return "delete-success";
* }
* @RequestMapping(value = "/deleteSubject", method = RequestMethod.GET)
* public String deleteSubject(Model model) {
* System.out.println("delete Book");
* return "delete-subject";
* }
* @RequestMapping(value = "/performDeleteSubject", method = RequestMethod.GET)
* public String performDeleteSubject(@RequestParam("subjectId") String subjectId, Model model) {
* System.out.println("delete subjectId " + subjectId);
* boolean isSuccess = subjectService.deleteSubject(Long.parseLong(subjectId));
* model.addAttribute("ENTITY\_NAME", "Subject");
* model.addAttribute("isSuccess", isSuccess);
* model.addAttribute("ENTITY\_ID", subjectId);
* return "delete-success";
* }
* @RequestMapping(value = "/search", method = RequestMethod.GET)
* public String search(Model model) {
* System.out.println("search Subject");
* return "search";
* }
* @RequestMapping(value = "/performSearch", method = RequestMethod.GET)
* public String performSearch(@RequestParam("id") String id, @RequestParam("entityName") String entityName,
* Model model) {
* System.out.println(entityName + "Perform Search" + id);
* Subject subject = null;
* Book book = null;
* String name = "";
* if (entityName.equals("subject")) {
* subject = subjectService.searchSubject(Long.parseLong(id));
* System.out.println("subject "+subject.getSubjectId());
* name = "subject";
* }
* if (entityName.equals("book")) {
* book = bookService.searchBook(Long.parseLong(id));
* name = "book";
* }
* model.addAttribute("ENTITY\_NAME", name);
* model.addAttribute("book", book);
* model.addAttribute("subject", subject);
* return "result";
* }
* @RequestMapping(value = "/exit", method = RequestMethod.GET)
* public String exit(Model model) {
* System.out.println("exit");
* return "exit";
* }
* }

# Book.java

* /\*\*
* \*
* \*/
* package com.cognizant.fsd.springtestassignment1.model;
* import java.io.Serializable;
* import java.time.LocalDate;
* import java.util.HashSet;
* import java.util.Set;
* import javax.persistence.CascadeType;
* import javax.persistence.Column;
* import javax.persistence.Entity;
* import javax.persistence.FetchType;
* import javax.persistence.GeneratedValue;
* import javax.persistence.GenerationType;
* import javax.persistence.Id;
* import javax.persistence.ManyToMany;
* import javax.persistence.Table;
* import javax.persistence.Transient;
* import org.springframework.format.annotation.DateTimeFormat;
* /\*\*
* \* @author Sudhir Kumar Thakur
* \*
* \*/
* @Entity
* @Table(name="book")
* public class Book implements Serializable{
* @Transient
* private static final long serialVersionUID = 3667779253735136971L;
* @Id
* //@GeneratedValue(strategy=GenerationType.AUTO)
* //private Long bid;
* @Column(name="BOOKID")
* private Long bookId;
* @Column(name="TITLE")
* private String title;
* @Column(name="PRICE")
* private Double price;
* @Column(name="VOLUME")
* private Integer volume;
* @Column(name="PUBLISHDATE")
* @DateTimeFormat(pattern = "yyyy-MM-dd")
* private LocalDate publishDate;
* @ManyToMany(mappedBy = "references",fetch=FetchType.EAGER)
* private Set<Subject> references = new HashSet<>();
* public Long getBookId() {
* return bookId;
* }
* public void setBookId(Long bookId) {
* this.bookId = bookId;
* }
* public String getTitle() {
* return title;
* }
* public void setTitle(String title) {
* this.title = title;
* }
* public Double getPrice() {
* return price;
* }
* public void setPrice(Double price) {
* this.price = price;
* }
* public Integer getVolume() {
* return volume;
* }
* public void setVolume(Integer volume) {
* this.volume = volume;
* }
* public LocalDate getPublishDate() {
* return publishDate;
* }
* public void setPublishDate(LocalDate publishDate) {
* this.publishDate = publishDate;
* }
* public Set<Subject> getReferences() {
* return references;
* }
* public void setReferences(Set<Subject> references) {
* this.references = references;
* }
* @Override
* public String toString() {
* return "Book [bookId=" + bookId + ", title=" + title + ", price=" + price + ", volume=" + volume
* + ", publishDate=" + publishDate + "]";
* }
* }

# Subject.java

* package com.cognizant.fsd.springtestassignment1.model;
* import java.io.Serializable;
* import java.util.HashSet;
* import java.util.List;
* import java.util.Set;
* import javax.persistence.CascadeType;
* import javax.persistence.Column;
* import javax.persistence.Entity;
* import javax.persistence.FetchType;
* import javax.persistence.GeneratedValue;
* import javax.persistence.GenerationType;
* import javax.persistence.Id;
* import javax.persistence.ManyToMany;
* import javax.persistence.Table;
* import javax.persistence.Transient;
* /\*\*
* \* @author Sudhir Kumar Thakur
* \*
* \*/
* @Entity
* @Table(name="subject")
* public class Subject implements Serializable{
* @Transient
* private static final long serialVersionId = 1L;
* @Id
* //@GeneratedValue(strategy=GenerationType.AUTO)
* //private Long sid;
* @Column(name="SUBJECTID")
* private Long subjectId;
* @Column(name="SUBTITLE")
* private String subTitle;
* @Column(name="DURATIONINHOURS")
* private Integer durationInHours;
* @ManyToMany(fetch=FetchType.EAGER)
* private Set<Book> references=new HashSet<Book>();
* @Transient
* private List<Long> bookList;
* public Long getSubjectId() {
* return subjectId;
* }
* public void setSubjectId(Long subjectId) {
* this.subjectId = subjectId;
* }
* public String getSubTitle() {
* return subTitle;
* }
* public void setSubTitle(String subTitle) {
* this.subTitle = subTitle;
* }
* public Integer getDurationInHours() {
* return durationInHours;
* }
* public void setDurationInHours(Integer durationInHours) {
* this.durationInHours = durationInHours;
* }
* public Set<Book> getReferences() {
* return references;
* }
* public void setReferences(Set<Book> references) {
* this.references = references;
* }
* @Override
* public String toString() {
* return "Subject [subjectId=" + subjectId + ", subTitle=" + subTitle + ", durationInHours=" + durationInHours
* + ", references=" + references + "]";
* }
* public List<Long> getBookList() {
* return bookList;
* }
* public void setBookList(List<Long> bookList) {
* this.bookList = bookList;
* }
* }

# BookService.java

* package com.cognizant.fsd.springtestassignment1.service;
* import java.util.List;
* import com.cognizant.fsd.springtestassignment1.model.Book;
* public interface BookService {
* public Book addBook(Book book);
* public boolean deleteBook(long bookId);
* public Book searchBook(long bookId);
* public List<Book> fetchAllBook();
* }

# BookServiceImpl.java

* package com.cognizant.fsd.springtestassignment1.service;
* import java.util.List;
* import java.util.Optional;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.stereotype.Service;
* import org.springframework.transaction.annotation.Transactional;
* import com.cognizant.fsd.springtestassignment1.model.Book;
* import com.cognizant.fsd.springtestassignment1.repository.BookRepository;
* @Service("bookService")
* @Transactional
* public class BookServiceImpl implements BookService {
* @Autowired
* private BookRepository bookRepository;
* public void setBookRepository(BookRepository bookRepository) {
* this.bookRepository = bookRepository;
* }
* @Override
* public Book addBook(Book book) {
* return bookRepository.save(book);
* }
* @Override
* public boolean deleteBook(long bookId) {
* bookRepository.deleteById(bookId);
* return true;
* }
* @Override
* public Book searchBook(long bookId) {
* Optional<Book> book=bookRepository.findById(bookId);
* return book!=null?book.get():null;
* }
* @Override
* public List<Book> fetchAllBook() {
* return (List<Book>) bookRepository.findAll();
* }
* }

# SubjectService.java

* package com.cognizant.fsd.springtestassignment1.service;
* import java.util.List;
* import com.cognizant.fsd.springtestassignment1.model.Subject;
* public interface SubjectService {
* public Subject addSubject(Subject subject);
* public boolean deleteSubject(long subjectId);
* public Subject searchSubject(long subjectId);
* public List<Subject> fetchAllSubject();
* }

# SubjectServiceImpl.java

* package com.cognizant.fsd.springtestassignment1.service;
* import java.util.List;
* import java.util.Optional;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.stereotype.Service;
* import org.springframework.transaction.annotation.Transactional;
* import com.cognizant.fsd.springtestassignment1.model.Subject;
* import com.cognizant.fsd.springtestassignment1.repository.SubjectRepository;
* @Service("subjectService")
* @Transactional
* public class SubjectServiceImpl implements SubjectService{
* @Autowired
* private SubjectRepository subjectRepository;
* public void setSubjectRepository(SubjectRepository subjectRepository) {
* this.subjectRepository = subjectRepository;
* }
* @Override
* public Subject addSubject(Subject subject) {
* return subjectRepository.save(subject);
* }
* @Override
* public boolean deleteSubject(long subjectId) {
* subjectRepository.deleteById(subjectId);
* return true;
* }
* @Override
* public Subject searchSubject(long subjectId) {
* Optional<Subject> subject=subjectRepository.findById(subjectId);
* return subject!=null?subject.get():null;
* }
* @Override
* public List<Subject> fetchAllSubject() {
* return (List<Subject>) subjectRepository.findAll();
* }
* }

# AbstractDao.java

* package com.cognizant.fsd.springtestassignment1.repository;
* import org.hibernate.Session;
* import org.hibernate.SessionFactory;
* import org.springframework.beans.factory.annotation.Autowired;
* public abstract class AbstractDao {
* @Autowired
* private SessionFactory sessionFactory;
* protected Session getSession() {
* return sessionFactory.getCurrentSession();
* }
* public void persist(Object entity) {
* getSession().saveOrUpdate(entity);
* }
* public void delete(Object entity) {
* getSession().delete(entity);
* }
* }

# BookRepository.java

* package com.cognizant.fsd.springtestassignment1.repository;
* import org.springframework.data.repository.CrudRepository;
* import com.cognizant.fsd.springtestassignment1.model.Book;
* public interface BookRepository extends CrudRepository<Book, Long>{
* }

# ApplicationTests.java

* package com.cognizant.fsd.springtestassignment1;
* import static org.mockito.Mockito.when;
* import java.util.ArrayList;
* import java.util.List;
* import java.util.Optional;
* import org.junit.Assert;
* import org.junit.Test;
* import org.junit.runner.RunWith;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.boot.test.context.SpringBootTest;
* import org.springframework.boot.test.mock.mockito.MockBean;
* import org.springframework.test.context.junit4.SpringRunner;
* import com.cognizant.fsd.springtestassignment1.model.Book;
* import com.cognizant.fsd.springtestassignment1.model.Subject;
* import com.cognizant.fsd.springtestassignment1.repository.BookRepository;
* import com.cognizant.fsd.springtestassignment1.repository.SubjectRepository;
* import com.cognizant.fsd.springtestassignment1.service.BookService;
* import com.cognizant.fsd.springtestassignment1.service.SubjectService;
* @RunWith(SpringRunner.class)
* @SpringBootTest
* public class ApplicationTests {
* @MockBean
* private SubjectRepository subjectRepository;
* @MockBean
* private BookRepository bookRepository;
* @Autowired
* private SubjectService subjectService;
* @Autowired
* private BookService bookService;
* //@Test
* public void bookServiceSearchSubjectTest() {
* Book book=new Book();
* book.setBookId(201l);
* book.setTitle("Java How To Program");
* Optional<Book> optionalBook = Optional.ofNullable(book);
* when(bookRepository.findById(201l)).thenReturn(optionalBook);
* //testing bookService.searchBook
* //Assert.assertEquals(book, bookService.searchBook(201l));
* Assert.assertNotEquals(book, bookService.searchBook(201l));
* }
* //@Test
* public void bookServiceAddSubjectTest() {
* Book book=new Book();
* book.setBookId(201l);
* book.setTitle("Java How To Program");
* when(bookRepository.save(book)).thenReturn(book);
* //testing bookService.searchBook
* //Assert.assertEquals(book, bookService.addBook(book));
* Assert.assertNotEquals(book, bookService.addBook(book));
* }
* @Test
* public void bookServiceFetchAllBookTest() {
* List<Book> bookList=new ArrayList<Book>();
* Book book1=new Book();
* book1.setBookId(201l);
* book1.setTitle("Spring In Action");
* Book book2=new Book();
* book2.setBookId(202l);
* book2.setTitle("Java How To Program");
* bookList.add(book1);
* bookList.add(book2);
* when(bookRepository.findAll()).thenReturn(bookList);
* //testing bookService.fetchAllBook
* //Assert.assertEquals(bookList, bookService.fetchAllBook());
* Assert.assertNotEquals(bookList, bookService.fetchAllBook());
* }


* //@Test
* public void subjectServiceFetchAllSubjectTest() {
* List<Subject> subjectList=new ArrayList<Subject>();
* Subject subjectObject = new Subject();
* subjectObject.setSubjectId(1l);
* subjectObject.setSubTitle("Bengali");
* Subject subject = new Subject();
* subject.setSubjectId(1l);
* subject.setSubTitle("Computer");
* subjectList.add(subjectObject);
* subjectList.add(subject);
* when(subjectRepository.findAll()).thenReturn(subjectList);
* //testing subjectService.fetchAllSubject
* //Assert.assertEquals(subjectList, subjectService.fetchAllSubject());
* Assert.assertNotSame(subjectList, subjectService.fetchAllSubject());
* }
* //@Test
* public void subjectServiceAddSubjectTest() {
* Subject subjectObject = new Subject();
* Subject subject = new Subject();
* subject.setSubjectId(1l);
* subject.setSubTitle("Computer");
* when(subjectRepository.save(subject)).thenReturn(subject);
* //testing subjectService.addSubject
* Assert.assertEquals(subjectObject, subjectService.addSubject(subject));
* }
* //@Test
* public void subjectServiceSearchSubjectTest() {
* Subject subject = new Subject();
* subject.setSubjectId(2l);
* Optional<Subject> optionalsubject = Optional.ofNullable(subject);
* when(subjectRepository.findById(1l)).thenReturn(optionalsubject);
* //testing subjectService.searchSubject
* Assert.assertEquals(1l, subjectService.searchSubject(1l).getSubjectId().longValue());
* }
* }

# application.properties

* server.port=9000
* spring.mvc.view.prefix=/WEB-INF/views/
* spring.mvc.view.suffix=.jsp

# mysql.properties

* # jdbc
* jdbc.driverClassName=com.mysql.jdbc.Driver
* jdbc.url=jdbc:mysql://localhost:3306/spring\_assignment\_14?createDatabaseIfNotExist=true
* jdbc.user=root
* jdbc.pass=pass@word1
* # hibernate
* hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
* hibernate.show\_sql=true
* hibernate.hbm2ddl.auto=update

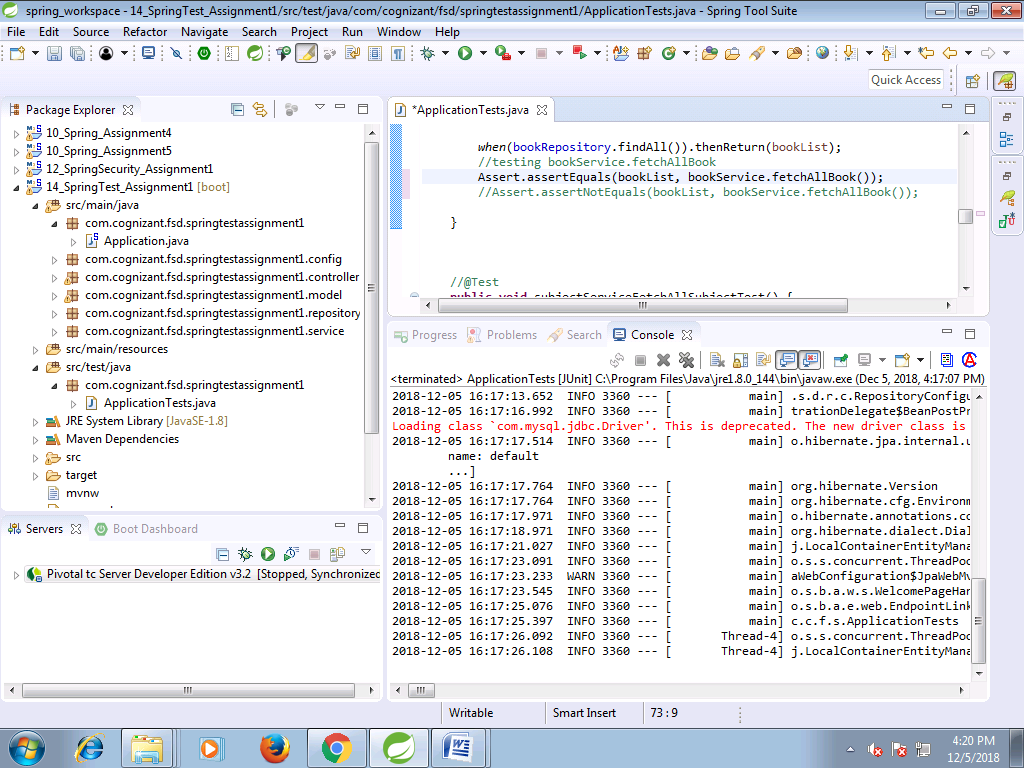
# pom.xml

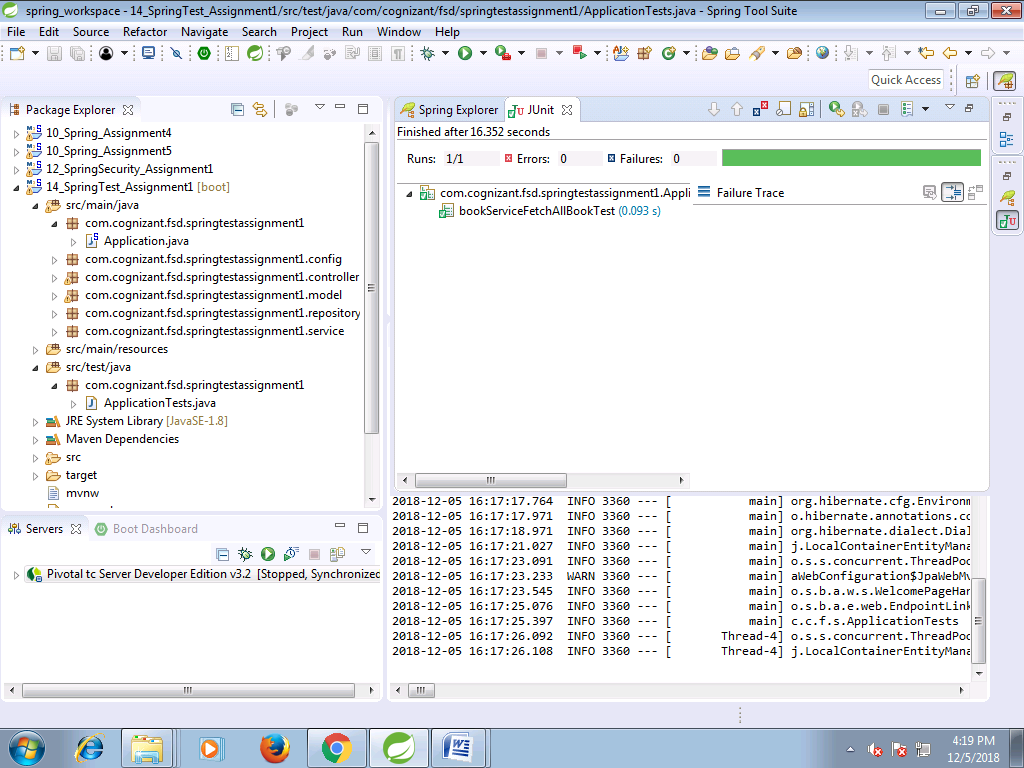
* <?xml version=*"1.0"* encoding=*"UTF-8"*?>
* <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*
* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>
* <modelVersion>4.0.0</modelVersion>
* <groupId>com.cognizant.fsd</groupId>
* <artifactId>13\_SpringBoot\_Assignment1</artifactId>
* <version>0.0.1-SNAPSHOT</version>
* <packaging>jar</packaging>
* <name>13\_SpringBoot\_Assignment1</name>
* <description>Demo project for Spring Boot</description>
* <parent>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-starter-parent</artifactId>
* <version>2.1.1.RELEASE</version>
* <relativePath /> <!-- lookup parent from repository -->
* </parent>
* <properties>
* <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
* <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
* <java.version>1.8</java.version>
* </properties>
* <dependencies>
* <dependency>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-starter-actuator</artifactId>
* </dependency>
* <dependency>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-starter-data-jpa</artifactId>
* </dependency>
* <!-- <dependency>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-starter-security</artifactId>
* </dependency> -->
* <dependency>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-starter-web</artifactId>
* </dependency>
* <dependency>
* <groupId>org.apache.tomcat.embed</groupId>
* <artifactId>tomcat-embed-jasper</artifactId>
* <scope>provided</scope>
* </dependency>
* <!-- JSTL -->
* <dependency>
* <groupId>javax.servlet</groupId>
* <artifactId>jstl</artifactId>
* </dependency>

* <dependency>
* <groupId>com.h2database</groupId>
* <artifactId>h2</artifactId>
* <scope>runtime</scope>
* </dependency>
* <dependency>
* <groupId>mysql</groupId>
* <artifactId>mysql-connector-java</artifactId>
* <scope>runtime</scope>
* </dependency>
* <dependency>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-starter-test</artifactId>
* <scope>test</scope>
* </dependency>
* <dependency>
* <groupId>org.springframework.security</groupId>
* <artifactId>spring-security-test</artifactId>
* <scope>test</scope>
* </dependency>
* </dependencies>
* <build>
* <plugins>
* <plugin>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-maven-plugin</artifactId>
* </plugin>
* </plugins>
* </build>
* </project>

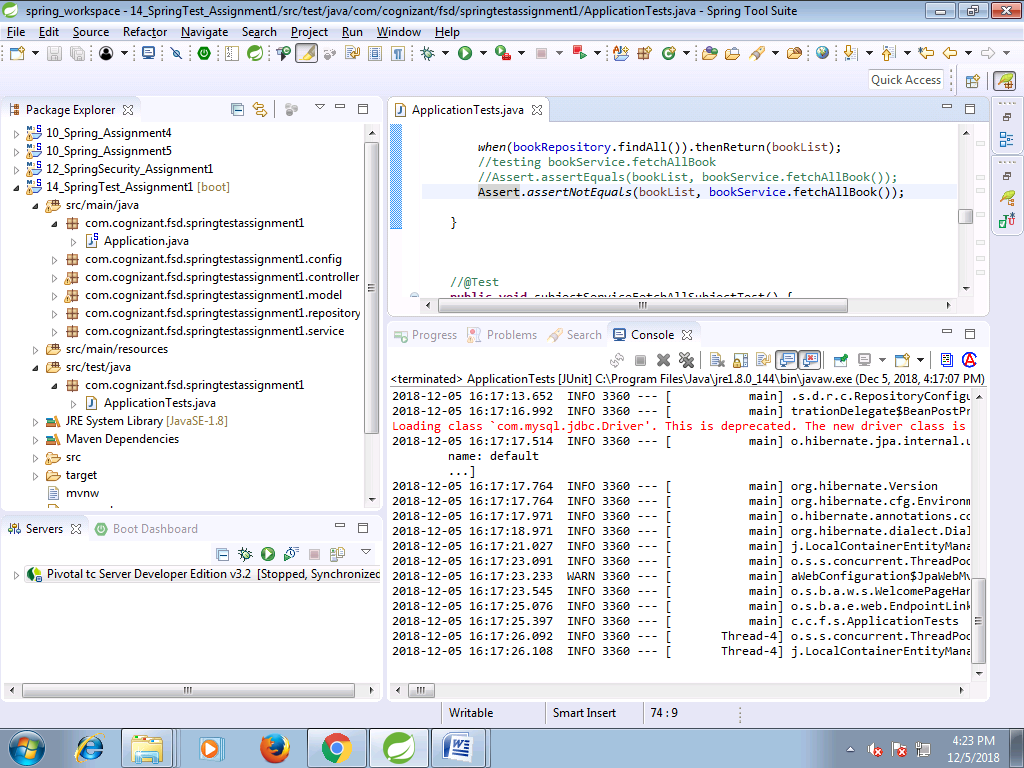
**Screenshots**

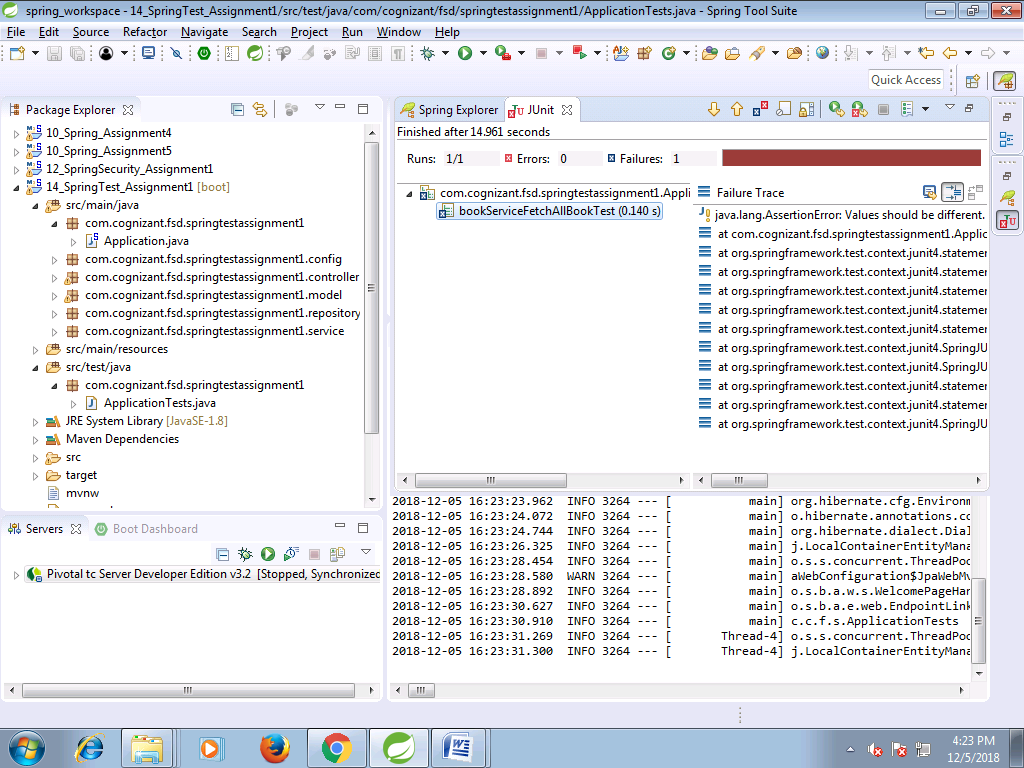
**Test case passed**

****

****

**Test case failed**

****

****