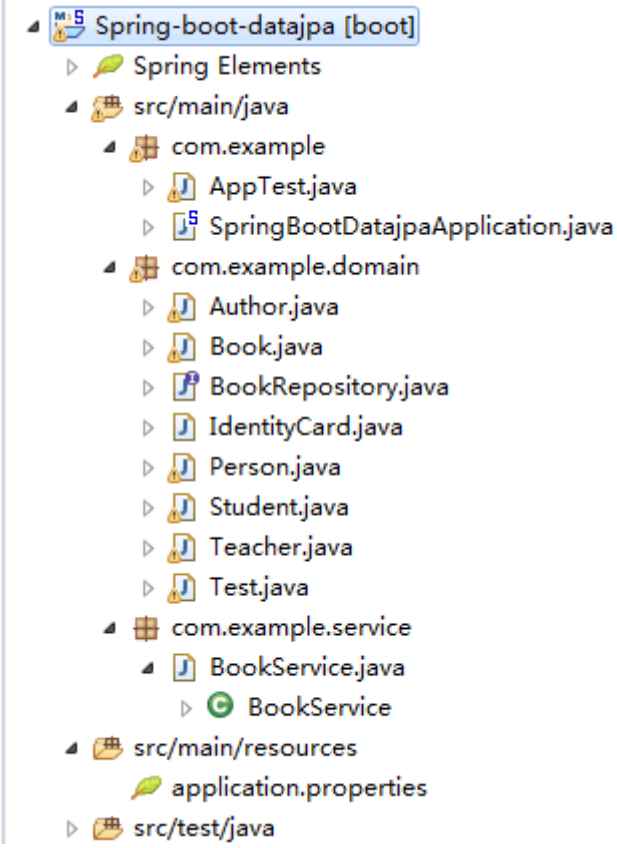
spring data jpa 项目

笔记本： 2018太极第一周作业笔记
创建时间： 2018/12/5 14:02 **更新时间：** 2018/12/5 14:06
作者： 2363655324idjzb

1.项目的文件



2.Book.java 文件

```
package com.example.domain;

import java.io.Serializable;
import java.math.BigDecimal;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OrderBy;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;

import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.CascadeType;
import org.hibernate.annotations.Where;

@Entity
@Table(name = "Book")
public class Book implements Serializable {
    private Integer id;
    private String bookName;
    private Float price;
    private Author author;
```

```
    private Integer flag;

    public Book() {
    }

    public Book(String bookName, Float price) {
        this.bookName = bookName;
        this.price = price;
    }

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "generator")
    @SequenceGenerator(sequenceName = "Book_SEQ", name = "generator", initialValue = 1, allocationSize = 1)
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    @Column(name = "bookName", length = 30)
    public String getBookName() {
        return bookName;
    }

    public void setBookName(String bookName) {
        this.bookName = bookName;
    }

    @Column(name = "price", columnDefinition = "decimal(4,2)")
    public Float getPrice() {
        return price;
    }

    public void setPrice(Float price) {
        this.price = price;
    }

    @ManyToOne
    @Where(clause = "flag = 1")
    @OrderBy("id asc")
    @JoinColumn(name = "author_id")
    @Cascade(CascadeType.SAVE_UPDATE)
    public Author getAuthor() {
        return author;
    }

    public void setAuthor(Author author) {
        this.author = author;
    }

    public Integer getFlag() {
        return flag;
    }

    public void setFlag(Integer flag) {
        this.flag = flag;
    }

    @Override
    public String toString() {
        return "Book [id=" + id + ", bookName=" + bookName + ", price=" + price + ", author=" + author +
"]";
    }

}
```

3.BookRepository.java 文件

```
package com.example.domain;
import java.util.List;
import  org.springframework.data.jpa.repository.JpaRepository;
import  org.springframework.data.jpa.repository.JpaSpecificationExecutor;
import  org.springframework.data.jpa.repository.Modifying;
import  org.springframework.data.jpa.repository.Query;
import  org.springframework.data.repository.query.Param;
public interface BookRepository extends  JpaRepository<Book, Integer>,  JpaSpecificationExecutor<Book> {
    // private Integer id;
    // private String bookName;
    // private Float price;
    // private Author author;
    @Query("select  b from Book b where  b.flag=1")
    List<Book> findAll();
    @Query("select b from Book b where  b.price=:price")
    List<Book> findByPrice(Float price);
    @Query("select b from Book b where  b.author=:author")
    List<Book> findByAuthor(Author author);
    @Query("select b from Book b where  b.id=:id")
    Book findByIdIs(Integer id);
    List<Book> findAllByIdNotNull();
    @Query("select b from Book b where  b.bookName=:%bookName%")
    List<Book> findAllByBookNameLike(String  bookName);
    List<Book>  findAllByIdGreaterThanEqualOrderByBookNameDesc(Integer start);
    @Query("select b from Book b where b.parent  is null and b.flag=1  order by b.id")
    List<Book> findRoots();
    @Query("select b from Book b where  b.id=:id")
    Book findById(@Param("id") Integer id);
    @Modifying
    @Query("update Book b set b.flag=0 where  b.id=:id")
    void updateFlag(@Param("id") String id);
    @Modifying
    @Query(value = "delete from Book b where  b.id=?1 and bookName=?2)", nativeQuery = true)
    void delete(@Param("id") Integer id,  @Param("bookName") String bookName);
}
```

4.BookService.java 文件

```
package com.example.service;
import java.util.ArrayList;
import java.util.List;
import javax.inject.Inject;
import  javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Predicate;
import javax.persistence.criteria.Root;
import org.springframework.data.domain.Page;
import  org.springframework.data.domain.PageRequest;
import  org.springframework.data.jpa.domain.Specification;
import  org.springframework.transaction.annotation.Propagation;
import  org.springframework.transaction.annotation.Transactional;
import com.example.domain.Author;
import com.example.domain.Book;
import com.example.domain.BookRepository;
public class BookService {
    @Inject
    BookRepository BookRepository;
    // 分页查询
    @Transactional(propagation =  Propagation.SUPPORTS)
    public List<Book> getBookPage() {
        PageRequest pageable = new  PageRequest(1, 6);
        Specification<Book> spec = new  Specification<Book>() {
            @Override
            public Predicate  toPredicate(Root<Book> root, CriteriaQuery<?>  criteriaQuery,
                    CriteriaBuilder  criteriaBuilder) {
                List<Predicate> list = new  ArrayList<>();
```

```
                return  criteriaBuilder.and(list.toArray(new  Predicate[0]));
            }
        };
        Page<Book> pageList =  BookRepository.findAll(spec, pageable);
        return pageList.getContent();
    }
    // 按照id查询book
    public  Book findByIdIs(Integer id) {
        Book book =  this.BookRepository.findOne(id);
        return book;
    }
    // 按照author查询book
    public Book findAuthor(Author author) {
        Book book = (Book)  this.BookRepository.findByAuthor(author);
        return book;
    }
    // 查询book
    @Transactional(propagation =  Propagation.SUPPORTS)
    public List<Book> findAllBook() {
        List<Book> book =  this.BookRepository.findAll();
        return book;
    }
    // 保存book
    @Transactional(propagation =  Propagation.REQUIRED)
    public void saveBook(Book book) {
        this.BookRepository.saveAndFlush(book);
    }
    // 删除book
        @Transactional(propagation =  Propagation.REQUIRED)
        public void deleteBook(Book book) {
            this.BookRepository.delete(book);
        }
}
```

4.测试

```
package com.example;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.inject.Inject;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;
import javax.persistence.TypedQuery;
import  javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import org.junit.Ignore;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import  org.springframework.boot.test.context.SpringBootTest;
import  org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import com.example.domain.Author;
import com.example.domain.Book;
import com.example.service.BookService;
@RunWith(SpringJUnit4ClassRunner.class)
@SpringBootTest
public class AppTest {
    private static final Logger log =  LoggerFactory.getLogger(AppTest.class);
    @Inject
    BookService bookService;
    //  Book
    //  Integer id;
    //  String bookName;
    //  Float price;
    //  Author author;
```

```
    @Ignore
    @Test
    public void saveTest() {
        Book book = bookService.findByIdIs(1);
        Author author = new Author();
        author.setAuthorName("wang");
        book.setBookName("aaa");
        book.setPrice((float) 12.30);
        book.setAuthor(author);
        book.setFlag(1);
        bookService.saveBook(book);
        List<Book> list =  bookService.findAllBook();
        System.out.println(list.size());
    }
    @Ignore
    @Test
    public void test2() {
        Book book = bookService.findByIdIs(1);
        System.out.println(book + "----book");
    }
    @PersistenceContext
    EntityManager em;
    @Ignore
    @Test
    public void test3() {
        CriteriaBuilder cb =  em.getCriteriaBuilder();
        CriteriaQuery<Book> c =  cb.createQuery(Book.class);
        Root<Book> book = c.from(Book.class);
        CriteriaQuery query =  c.select(book).where(cb.equal(book.get("bookName"), "asd"));
        TypedQuery query1 =  em.createQuery(query);
        List<Book> list =  query1.getResultList();
        System.out.println(query1.getResultList());
    }
    @Ignore
    @Test
    public void getBookPageTest() {
        List<Book> book =  bookService.getBookPage();
        System.out.println(book);
    }
    @Ignore
    @Test
    public void deleteTest(Book book) {
        bookService.deleteBook(book);
    }
}
```

5.配置 application.properties

```
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/jpa-2?
characterEncoding=utf8&useSSL=true&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=root
```

6.pom.文件

```
<?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/4.0.0http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>Spring-boot-datajpa</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>
    <name>Spring-boot-datajpa</name>
    <description>Demo project for Spring  Boot</description>
    <parent>
        <groupId>org.springframework.boot</groupId>
```

```
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.18.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-data-jpa</artifactId>
        </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>javax.inject</groupId>
            <artifactId>javax.inject</artifactId>
            <version>1</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
            <version>3.6</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>

        </dependency>
        <dependency>
            <groupId>org.apache.struts</groupId>
            <artifactId>struts2-convention-plugin</artifactId>
            <version>2.5.14.1</version>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```