**实验十 SpringBoot框架使用**

学生姓名： 张小锋 学 号： 8002119102 专业班级： 计软192班

实验类型：□ 验证 □ 综合 □ 设计 □ 创新 实验日期： 2021.12.11 实验成绩：

1. 实验项目名称

SpringBoot框架使用

1. 实验目的

1. 了解SpringBoot项目创建；

2. 了解JPA使用方法。

1. 实验基本原理

SpringBoot框架合成JPA（Java持久层API）操作Mysql数据库

1. 主要仪器设备及耗材

计算机、Idea、Tomcat9.0、JDK1.8

五、实验步骤（完整内容见光盘）

1.创建SpringBoot项目

（1）通过IDEA创建项目

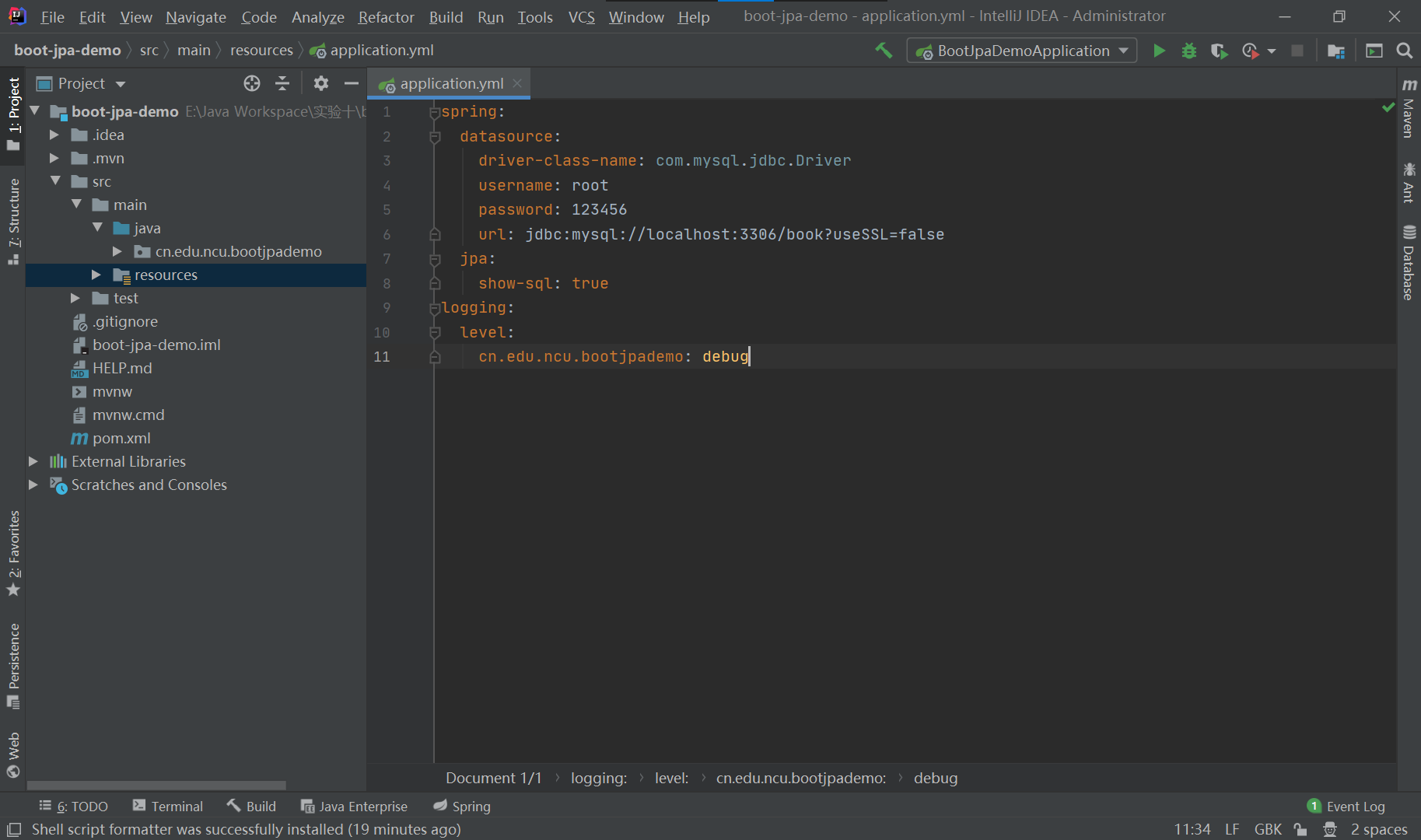
（2）选择需要的依赖

（3）进行项目相关配置

i. 修改配置文件类型为yml

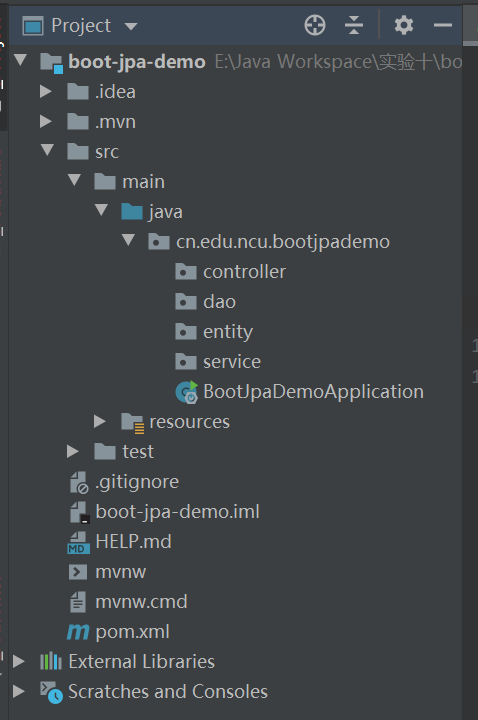
ii. 配置项目相关信息：数据库连接信息、JPA SQL语句输出、日志打印级别；也可以在这设置项目端口、路径等

创建成功的项目如下所示：



2.编码实现

（1）创建相应的package：entity、dao、service、controller



（2）书籍分类查询功能

a. 按照表结构创建分类entity，并通过idea自动生成getter、setter及toString方法

b. 类标记注解为@Entity，并在主键（@Id）和自增字段（@GeneratedValue）上添加相应的注解

1. package cn.edu.ncu.bootjpademo.entity;
2. import javax.persistence.GeneratedValue;
3. import javax.persistence.GenerationType;
4. import javax.persistence.Id;
5. import java.math.BigInteger;
6. import java.sql.Timestamp;
7. import javax.persistence.Entity;
8. @Entity
9. public class BookCategory {
10. @Override
11. public String toString() {
12. return "BookCategory{" +
13. "categoryId=" + categoryId +
14. ", categoryName='" + categoryName + '\'' +
15. ", createTime=" + createTime +
16. ", updataTime=" + updataTime +
17. '}';
18. }
19. public BigInteger getCategoryId() {
20. return categoryId;
21. }
22. public void setCategoryId(BigInteger categoryId) {
23. this.categoryId = categoryId;
24. }
25. public String getCategoryName() {
26. return categoryName;
27. }
28. public void setCategoryName(String categoryName) {
29. this.categoryName = categoryName;
30. }
31. public Timestamp getCreateTime() {
32. return createTime;
33. }
34. public void setCreateTime(Timestamp createTime) {
35. this.createTime = createTime;
36. }
37. public Timestamp getUpdataTime() {
38. return updataTime;
39. }
40. public void setUpdataTime(Timestamp updataTime) {
41. this.updataTime = updataTime;
42. }
43. @Id
44. @GeneratedValue(strategy= GenerationType.IDENTITY)
45. private BigInteger categoryId;
46. private String categoryName;
47. private Timestamp createTime;
48. private Timestamp updataTime;
49. }

c. 创建dao接口，接口类 extends JpaRepository<T,ID>，T和ID分别为相应的实体和主键类型

1. package cn.edu.ncu.bootjpademo.dao;
2. import cn.edu.ncu.bootjpademo.entity.BookCategory;
3. import org.springframework.data.jpa.repository.JpaRepository;
4. import org.springframework.stereotype.Repository;
5. import java.math.BigInteger;
6. @Repository
7. public interface BookCategoryDao extends JpaRepository<BookCategory, BigInteger> {
8. }

d. 创建相应的service

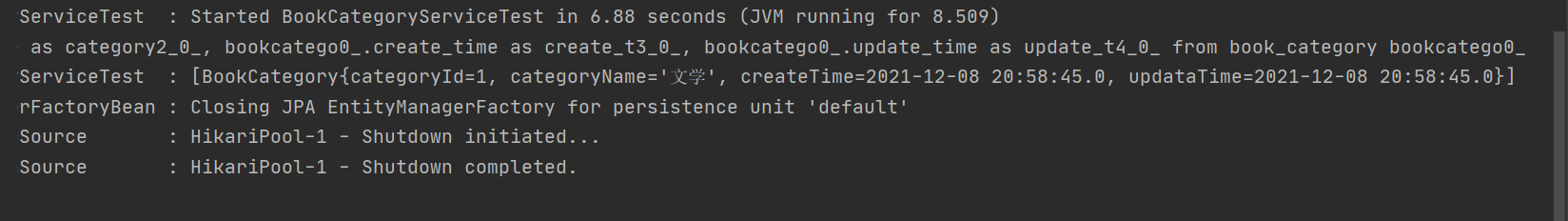
1. package cn.edu.ncu.bootjpademo.service;
2. import cn.edu.ncu.bootjpademo.dao.BookCategoryDao;
3. import cn.edu.ncu.bootjpademo.entity.BookCategory;
4. import org.springframework.beans.factory.annotation.Autowired;
5. import org.springframework.stereotype.Service;
6. import java.util.List;
7. @Service
8. public class BookCategoryService {
9. @Autowired
10. private BookCategoryDao bookCategoryDao;
11. */\*\**
12. @description 查询所有分类信息
13. @param
14. @return  List<BookCategory>
15. \*/
16. public List<BookCategory> findAll(){
17. return bookCategoryDao.findAll();
18. }
19. }

e. 创建service的单元测试方法

1. package cn.edu.ncu.bootjpademo.service;
2. import org.junit.Test;
3. import org.junit.runner.RunWith;
4. import org.slf4j.Logger;
5. import org.slf4j.LoggerFactory;
6. import org.springframework.beans.factory.annotation.Autowired;
7. import org.springframework.boot.test.context.SpringBootTest;
8. import org.springframework.test.context.junit4.SpringRunner;
9. @RunWith(SpringRunner.class)
10. @SpringBootTest
11. public class BookCategoryServiceTest {
12. private final Logger logger = LoggerFactory.getLogger(this.getClass());
13. @Autowired
14. private BookCategoryService service;
15. @Test
16. public void findAll() {
17. logger.debug(service.findAll().toString());
18. }
19. }

f. 运行测试方法，观察日志输入是否正常（可先在数据库表中添加测试数据）

预先在数据库表格中存入一行数据，运行测试方法，得到如下运行结果：



可以看到，数据库中的信息被成功打印了！

g. 创建controller类（\*\*BookCategoryController\*\*）及方法（\*\*findAl\*\*l）

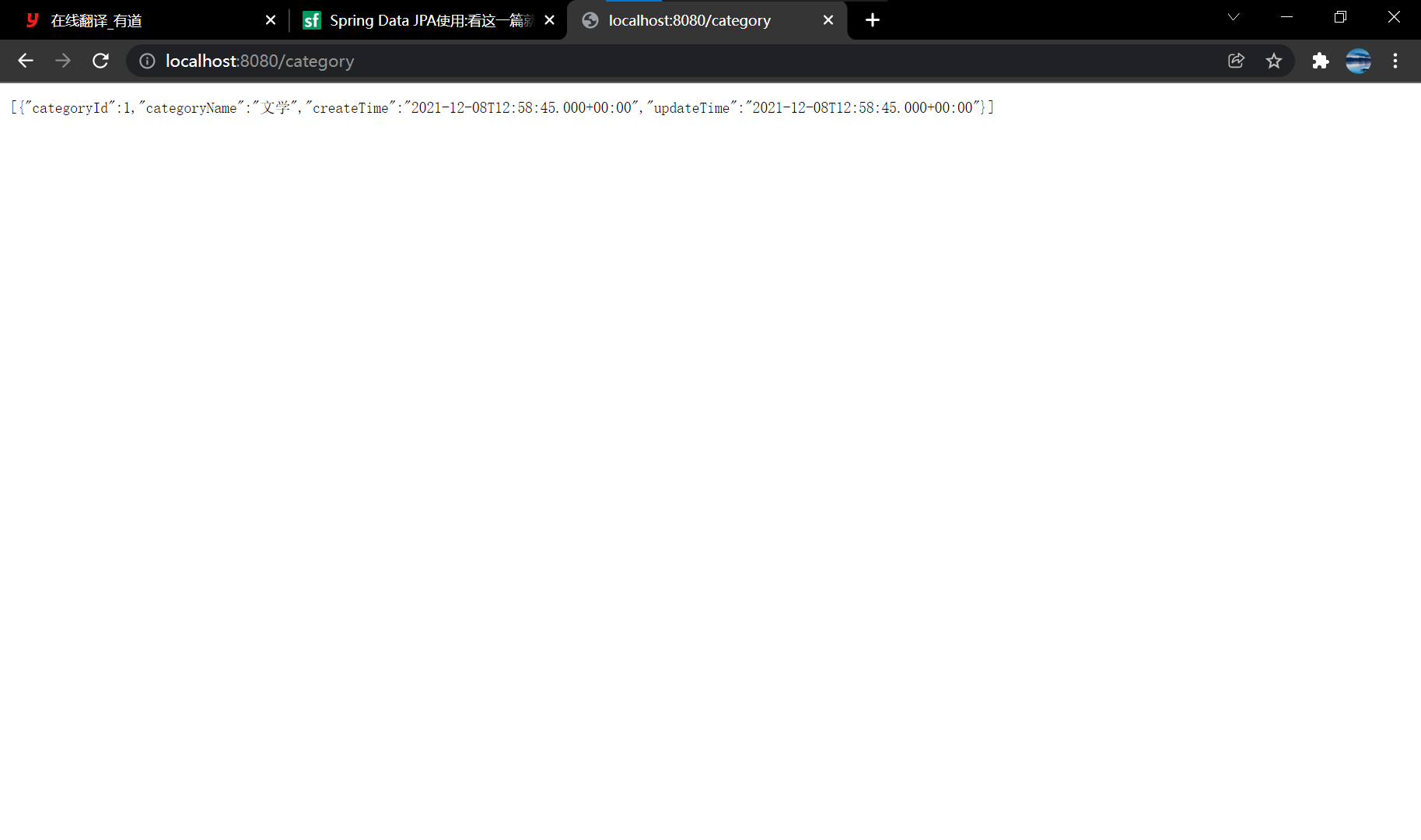
运行整个项目，并使用postman访问（get方法也可以通过浏览器访问）

Postman获取方式：(https://www.getpostman.com/ 或者 添加chrome插件方式获取)

controller类中的代码如下：

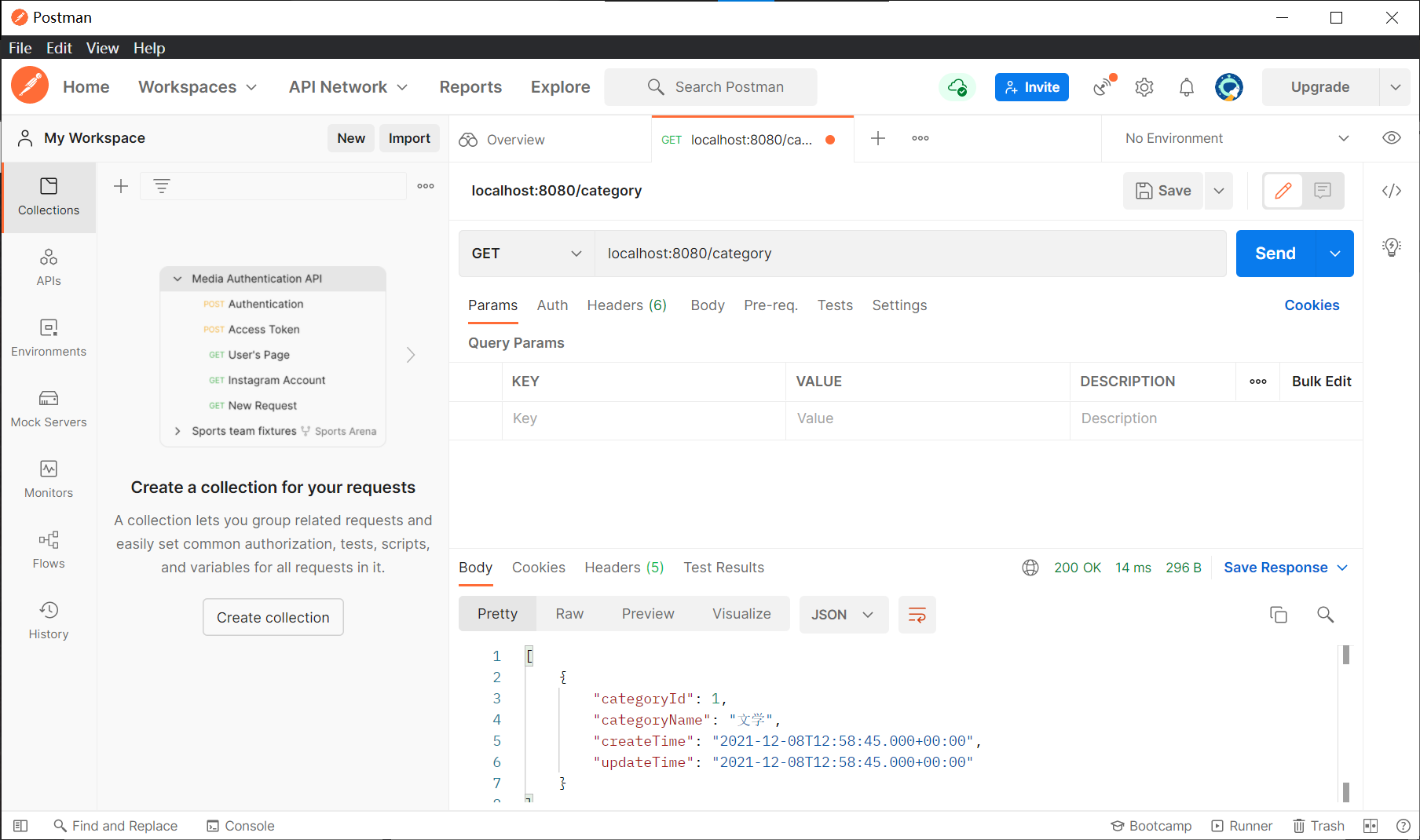
1. package cn.edu.ncu.bootjpademo.controller;
2. import cn.edu.ncu.bootjpademo.entity.BookCategory;
3. import cn.edu.ncu.bootjpademo.service.BookCategoryService;
4. import org.springframework.beans.factory.annotation.Autowired;
5. import org.springframework.web.bind.annotation.RequestMapping;
6. import org.springframework.web.bind.annotation.RequestMethod;
7. import org.springframework.web.bind.annotation.RestController;
8. import java.util.List;
9. @RestController
10. @RequestMapping(value = "category")
11. public class BookCategoryController {
12. @Autowired
13. private BookCategoryService service;
14. */\*\**
15. @Description 获取全部分类信息
16. @Param
17. @Return List<BookCategory>
18. \*/
19. @RequestMapping(value = "",method = RequestMethod.GET)
20. public List<BookCategory> findAll(){
21. return service.findAll();
22. }
23. }

直接通过浏览器访问结果如下：

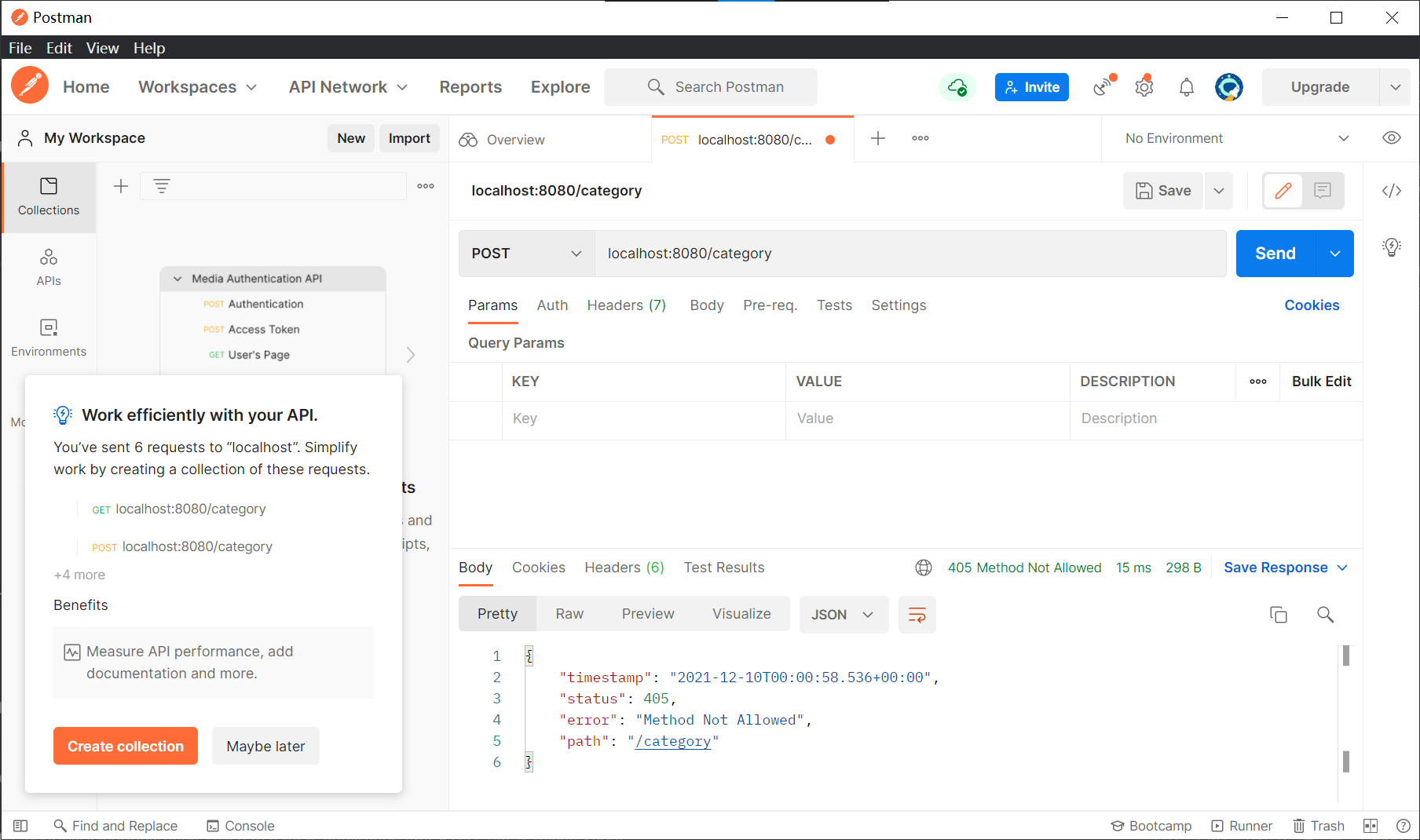


使用postman访问结果如下：

i．GET请求



ii. POST请求



可以看到由于没有编写POST请求对应的控制器，所以报405错误，提示错误信息

"Method Not Allowed"

（3）insert功能

a. 简单的CRUD功能JPA已经提供，不需要手工完成dao的实现类

b. dao不用修改，直接在service（BookServiceService）中添加插入功能（saveBookCategory）

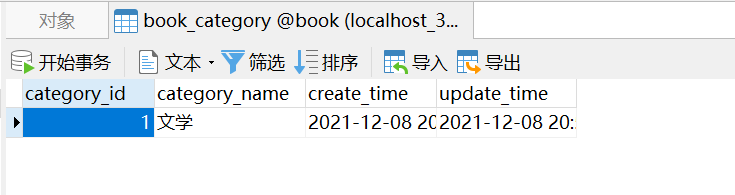
1. public void saveBookCategory(BookCategory bookCategory) {
2. bookCategoryDao.save(bookCategory);
3. }

c. 创建测试方法

因为JPA是懒加载，所以需要在测试类上加上注解@Transactional

1. @Test
2. public void save() {
3. BookCategory bookCategory = new BookCategory();
4. Timestamp t = new Timestamp(System.currentTimeMillis());
5. bookCategory.setCategoryName("科技");
6. bookCategory.setCreateTime(t);
7. bookCategory.setUpdateTime(t);
8. service.saveBookCategory(bookCategory);
9. }

d. 执行测试，查看结果（由于添加@Transactional 之后在测试类中并不会commit，所以数据库中不会有新增记录）



查看数据库可以知道，数据库中没有新增记录。

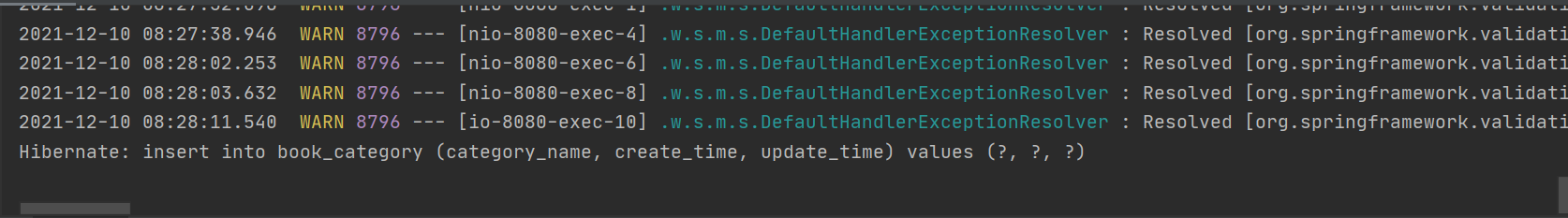
e. 添加contorller方法

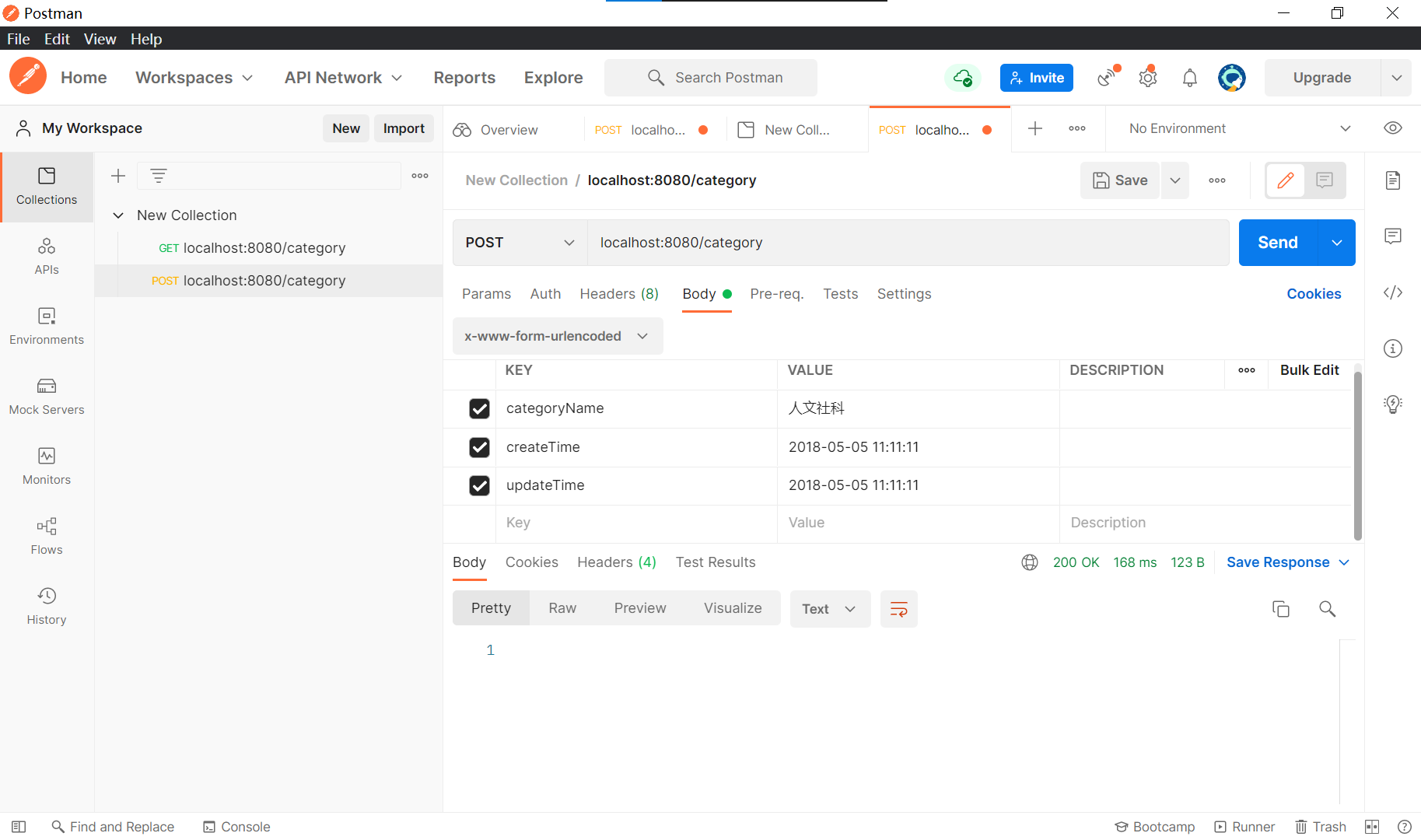
1. @PostMapping("")
2. public void saveBookCategory(BookCategory bookCategory){
3. service.saveBookCategory(bookCategory);
4. }

使用@PostMapping 等于 @RequestMapping(value,method = RequestMethod.POST)

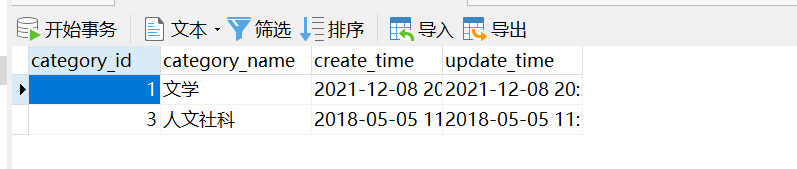
f. 启动项目，并使用Postman测试（时间格式为：yyyy-mm-dd hh:mm:ss）

查看控制台，显示成功插入了！





查看数据库中的数据，发现数据多了一行，成功插入！



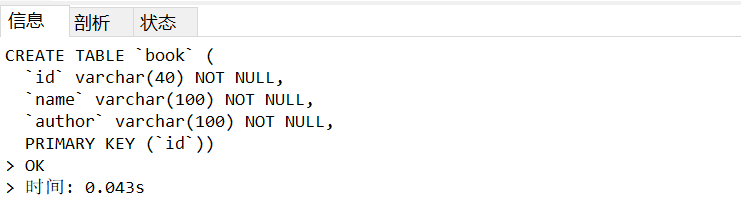
3. 手动映射及SQL

（1）表和字段

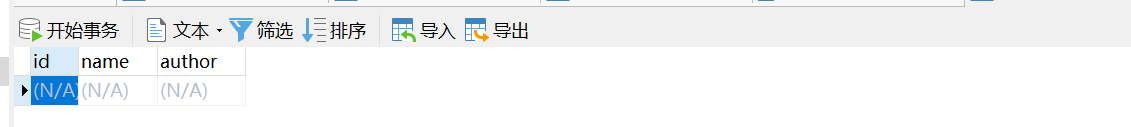
a. 建立一个非驼峰风格表

执行如下SQL语句创建表

1. CREATE TABLE `book` (
2. `id` varchar(40) NOT NULL,
3. `name` varchar(100) NOT NULL,
4. `author` varchar(100) NOT NULL,
5. PRIMARY KEY (`id`))



查看数据库中，成功创建的表如下：



b. 建立entity（记得生成getter、setter方法）

1. package cn.edu.ncu.bootjpademo.entity;
2. import javax.persistence.Column;
3. import javax.persistence.Entity;
4. import javax.persistence.Id;
5. import javax.persistence.Table;
6. @Entity
7. @Table(name = "book")
8. public class Book {
9. @Id
10. @Column(name = "id")
11. private String id;
12. @Column(name = "name")
13. private String name;
14. public String getId() {
15. return id;
16. }
17. public void setId(String id) {
18. this.id = id;
19. }
20. public String getName() {
21. return name;
22. }
23. public void setName(String name) {
24. this.name = name;
25. }
26. public String getAuthor() {
27. return author;
28. }
29. public void setAuthor(String author) {
30. this.author = author;
31. }
32. @Column(name = "author")
33. private String author;
34. }

c. 创建dao

1. package cn.edu.ncu.bootjpademo.dao;
2. import cn.edu.ncu.bootjpademo.entity.Book;
3. import org.springframework.data.jpa.repository.JpaRepository;
4. import org.springframework.data.jpa.repository.Query;
5. import org.springframework.stereotype.Repository;
6. import java.util.List;
7. @Repository
8. public interface BookDao extends JpaRepository<Book,String> {
9. @Query(value = "select \* from book where id < ?",nativeQuery = true)
10. List<Book> queryBook(String id);
11. }

e. 创建service

1. package cn.edu.ncu.bootjpademo.service;
2. import cn.edu.ncu.bootjpademo.dao.BookDao;
3. import cn.edu.ncu.bootjpademo.entity.Book;
4. import org.springframework.beans.factory.annotation.Autowired;
5. import org.springframework.stereotype.Service;
6. import java.util.List;
7. @Service
8. public class BookService {
9. @Autowired
10. private BookDao dao;
11. public List<Book> queryBook(String id){
12. return dao.queryBook(id);
13. }
14. }

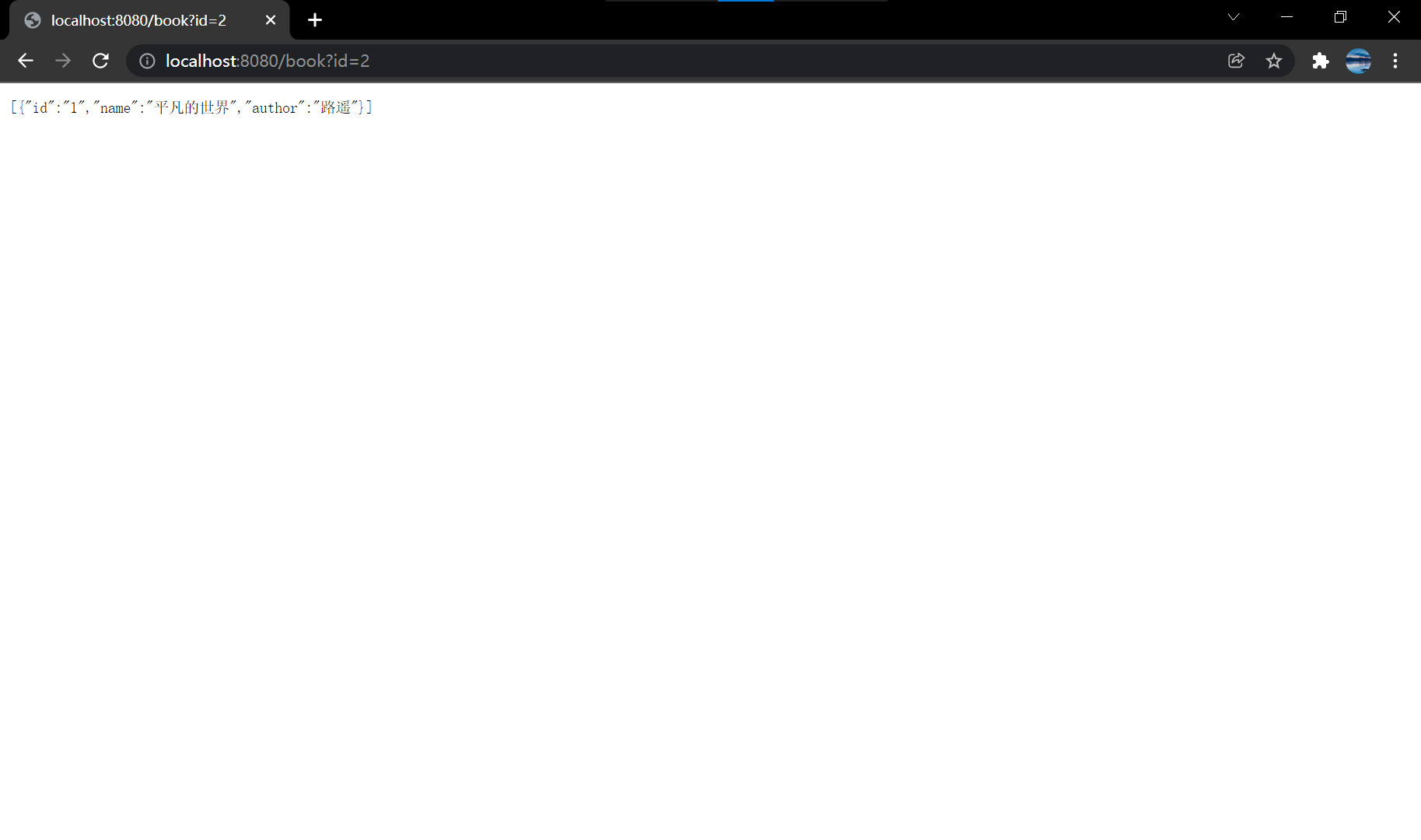
f. 进行单元测试，观察输出结果是否正确。

自己创建controller类及相应方法，并使用Postman或浏览器访问路径，并打印结果

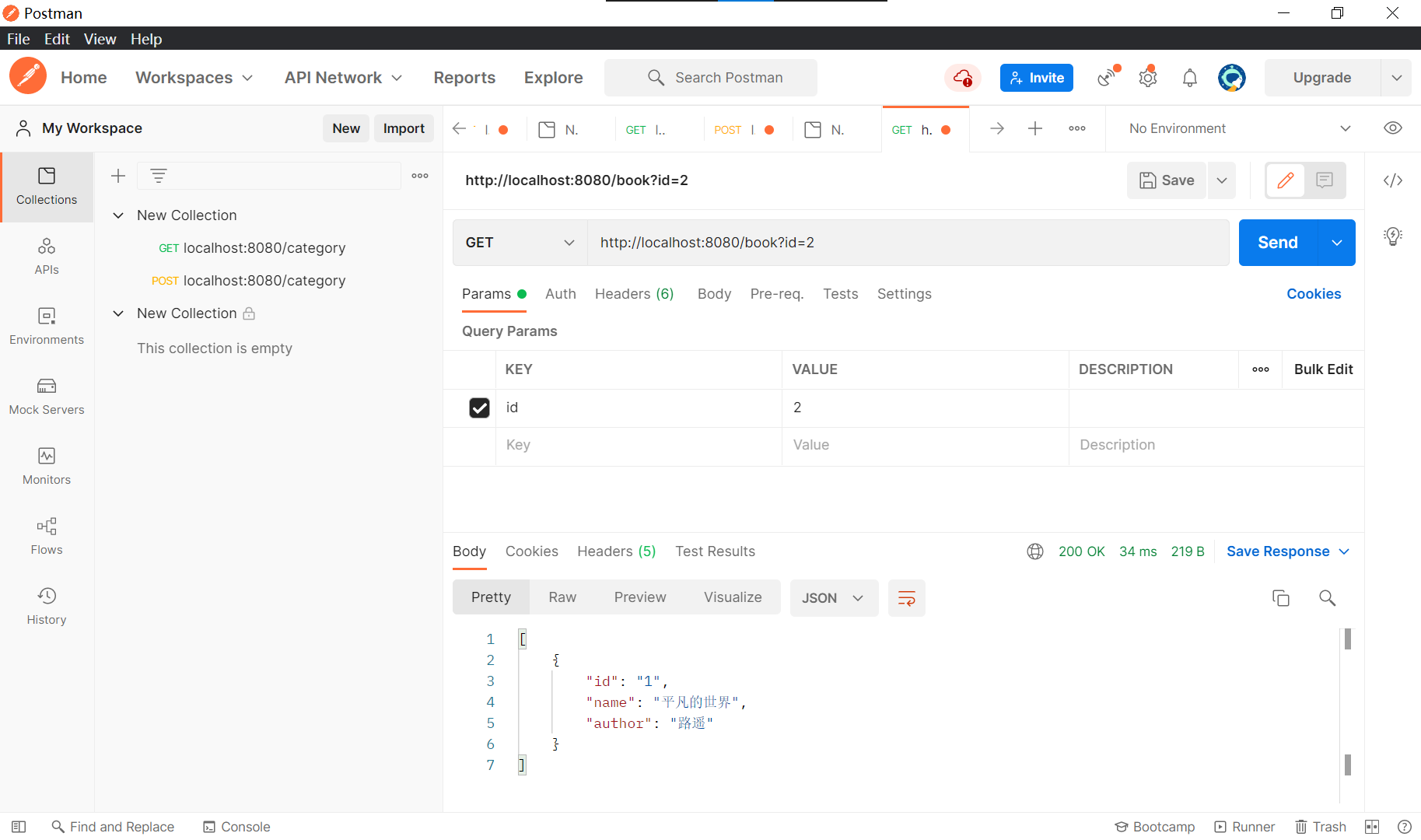
创建controller类及相应方法如下所示：

1. package cn.edu.ncu.bootjpademo.controller;
2. import cn.edu.ncu.bootjpademo.entity.Book;
3. import cn.edu.ncu.bootjpademo.service.BookService;
4. import org.springframework.beans.factory.annotation.Autowired;
5. import org.springframework.web.bind.annotation.GetMapping;
6. import org.springframework.web.bind.annotation.RequestMapping;
7. import org.springframework.web.bind.annotation.RequestParam;
8. import org.springframework.web.bind.annotation.RestController;
9. import java.util.List;
10. @RestController
11. @RequestMapping("/book")
12. public class BookController {
13. @Autowired
14. BookService bookService;
15. @GetMapping("")
16. public List<Book> queryBook(@RequestParam String id){
17. return bookService.queryBook(id);
18. }
19. }

使用浏览器访问：

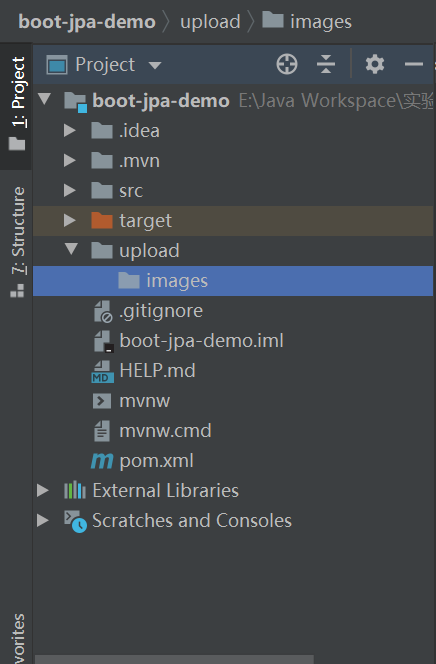


使用postman访问：



4.文件上传

（1）建立上传文件夹，建立 upload/images 文件夹

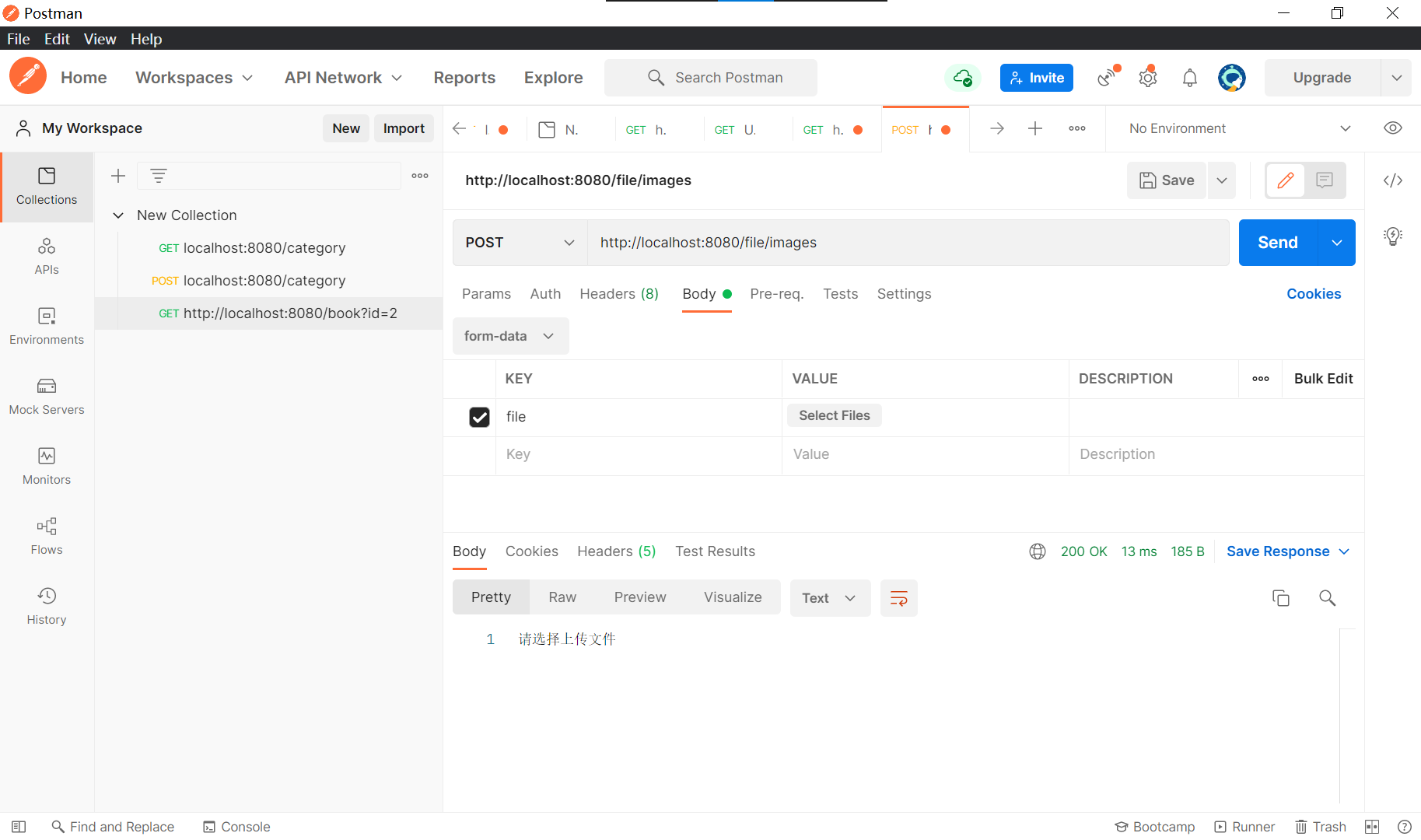


2. 在controller package下创建 UploadFileController类

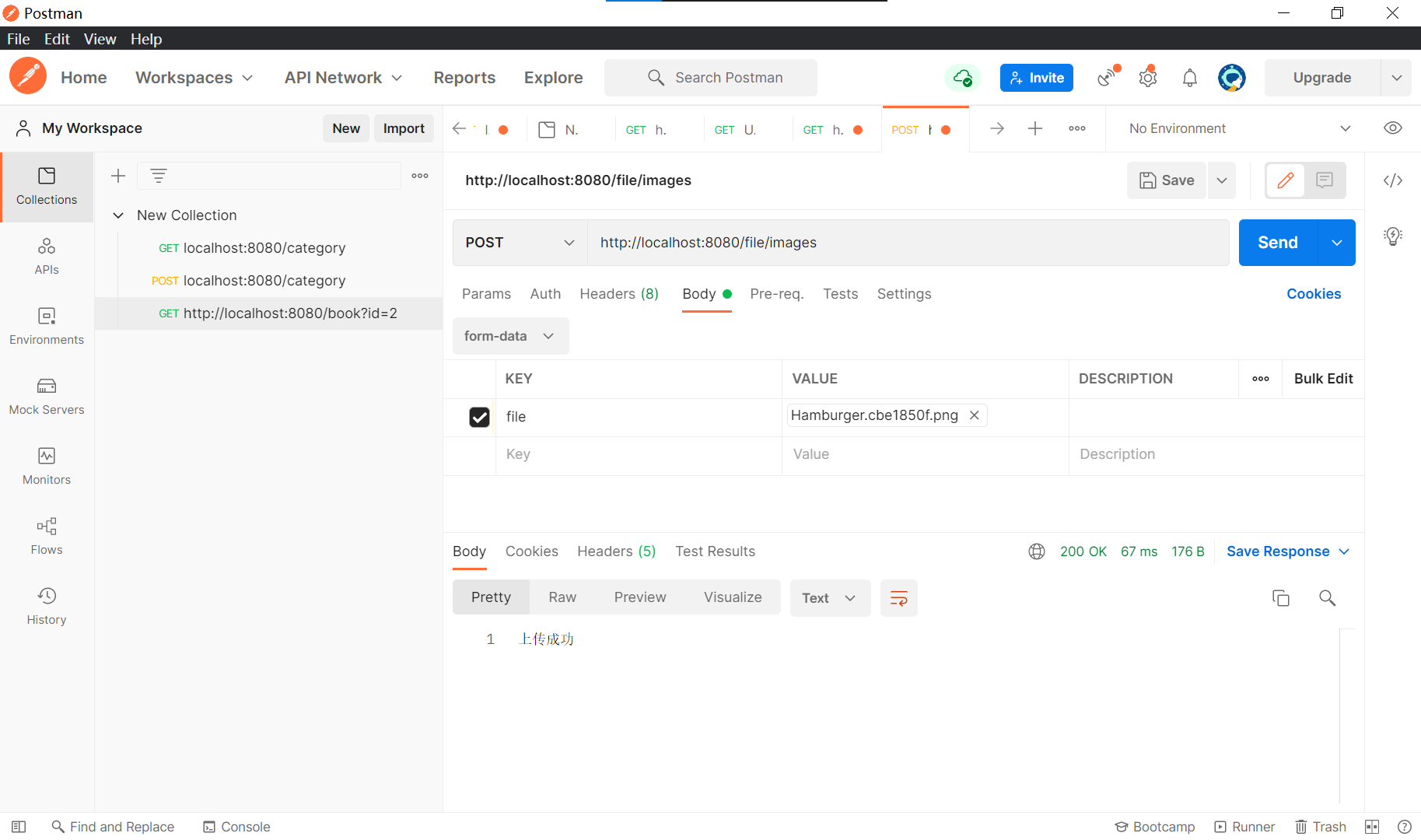
1. package cn.edu.ncu.bootjpademo.controller;
2. import org.springframework.web.bind.annotation.PostMapping;
3. import org.springframework.web.bind.annotation.RequestMapping;
4. import org.springframework.web.bind.annotation.RequestParam;
5. import org.springframework.web.bind.annotation.RestController;
6. import org.springframework.web.multipart.MultipartFile;
7. import java.io.IOException;
8. import java.nio.file.Files;
9. import java.nio.file.Path;
10. import java.nio.file.Paths;
11. @RestController
12. @RequestMapping("/file")
13. public class UploadFileController {
14. @PostMapping("/images")
15. public String singleFileUpload(@RequestParam("file") MultipartFile file) {
16. if (file.isEmpty()) {
17. return "请选择上传文件";
18. }
19. try {
20. byte[] bytes = file.getBytes();
21. Path path = Paths.get("upload/images/"+file.getOriginalFilename());
22. Files.write(path, bytes);
23. } catch (IOException e) {
24. e.printStackTrace();
25. }
26. return "上传成功";
27. }
28. }

3. 启动项目，使用Postman进行测试

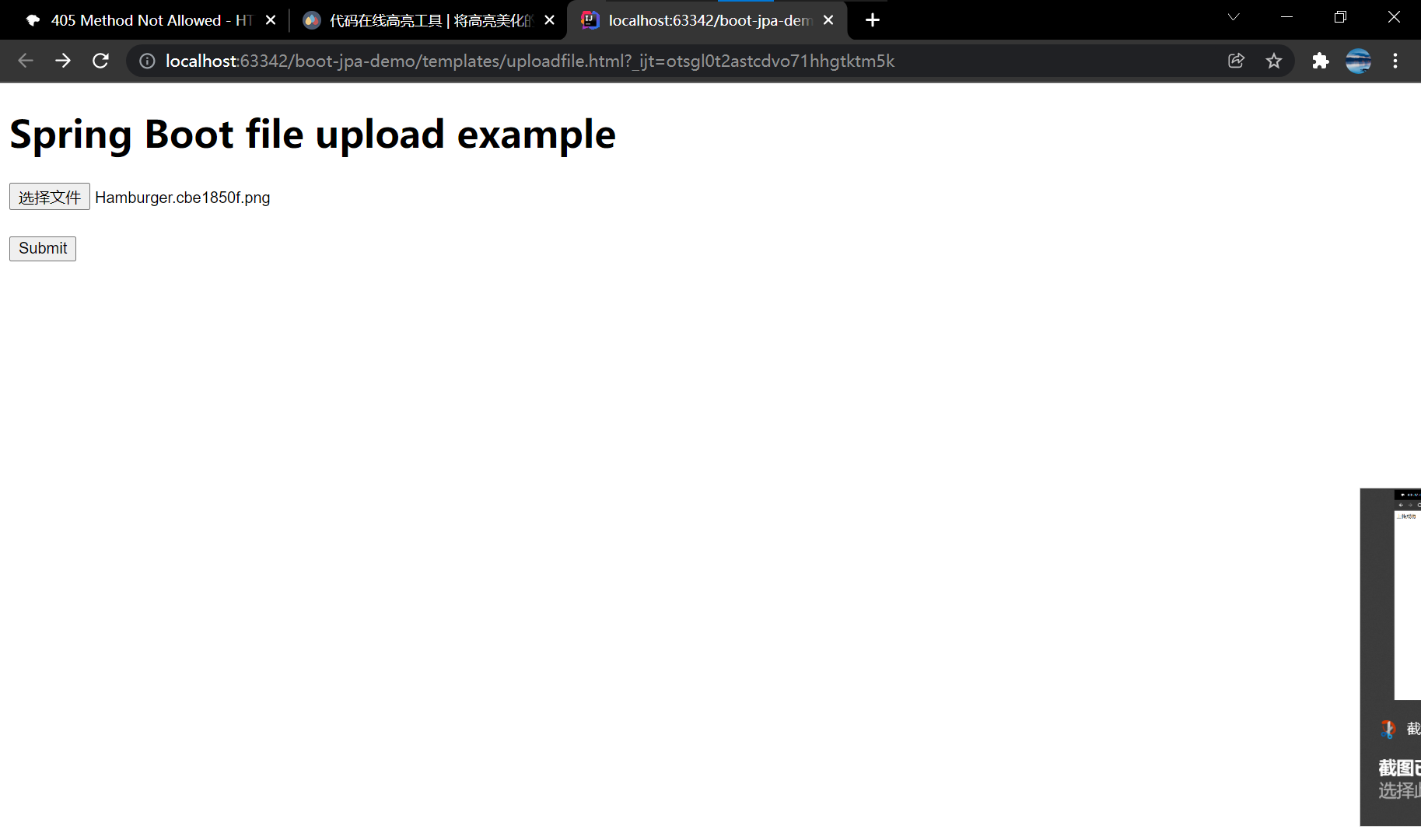
不选择文件时，请求结果如下：

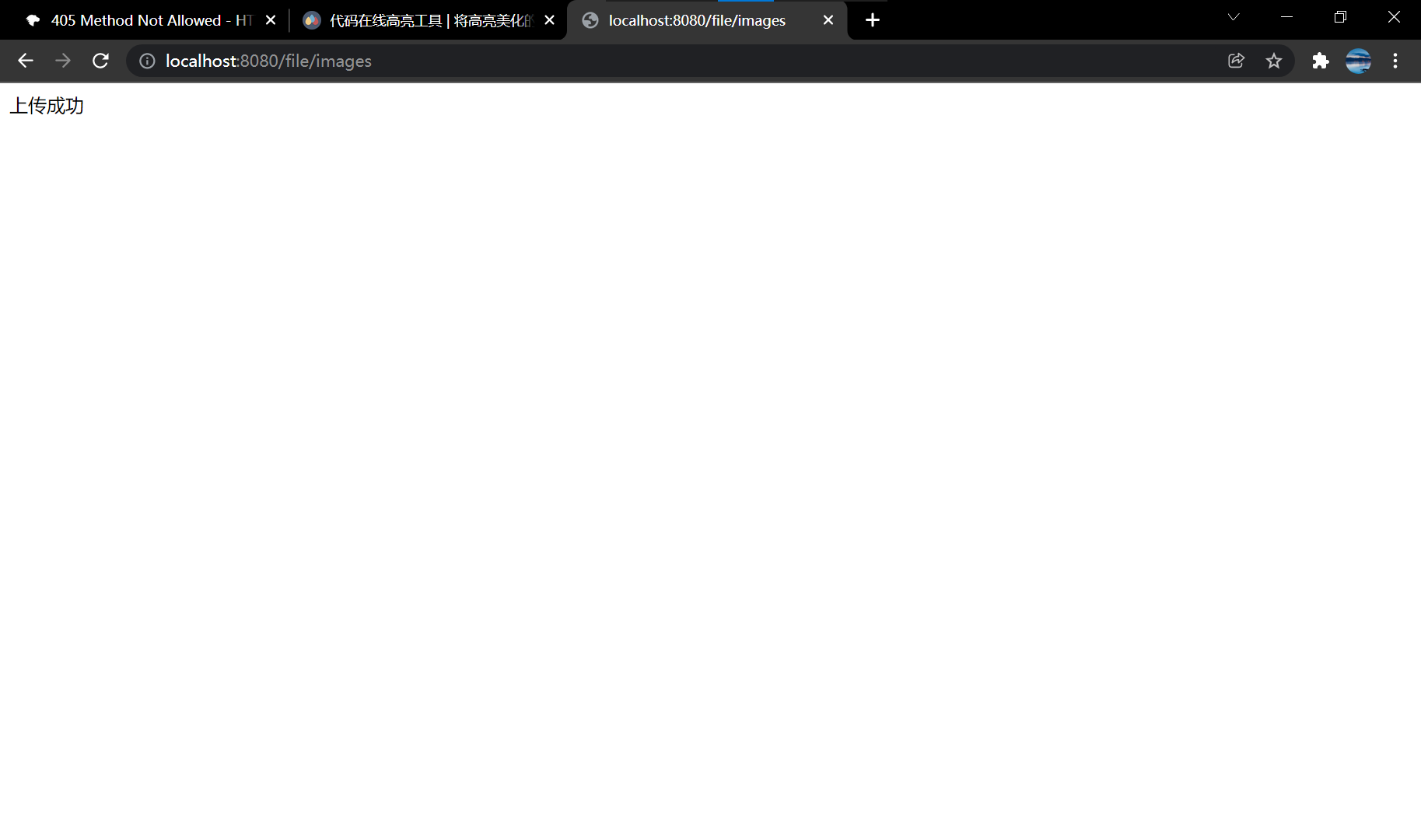


选择文件时，请求结果如下：

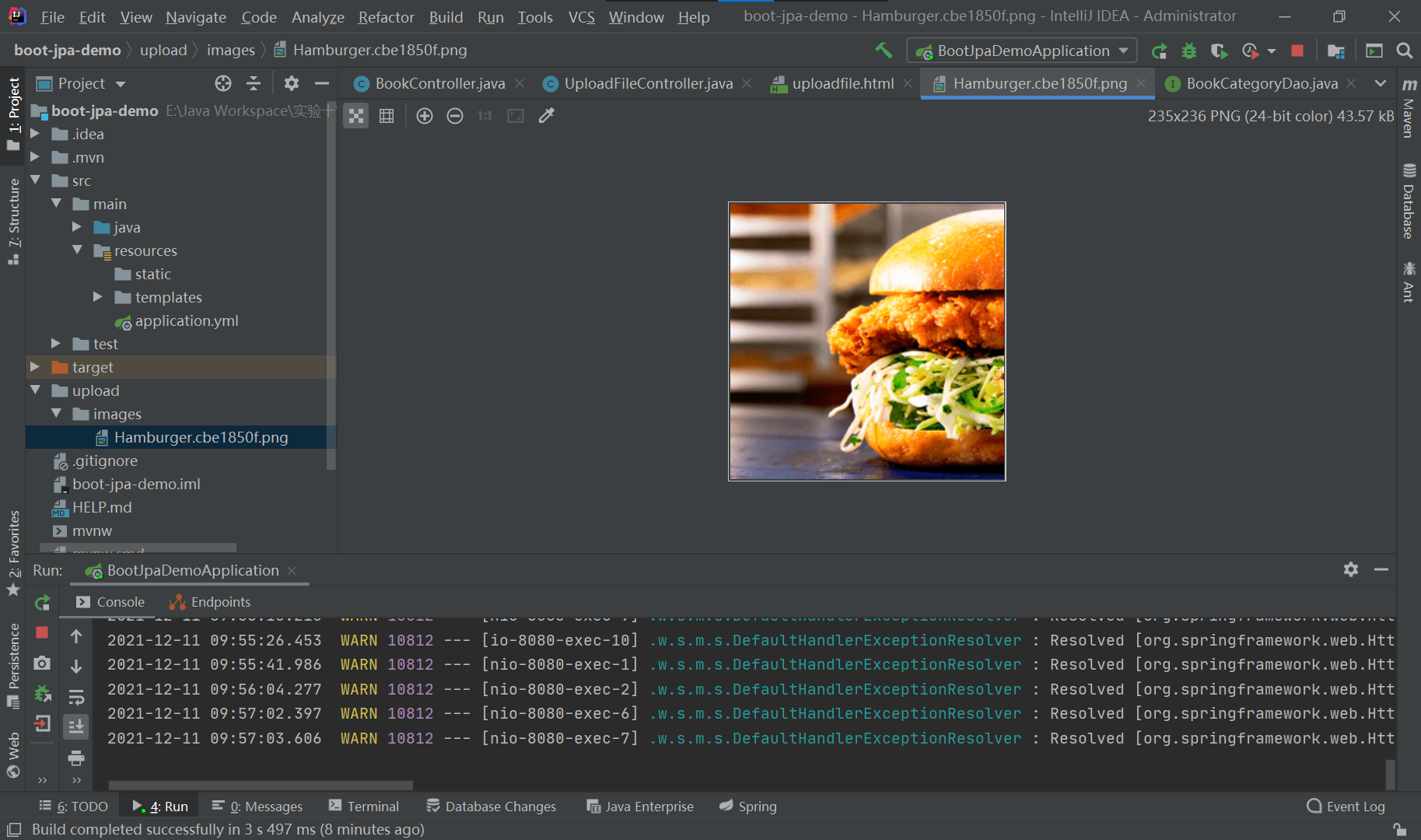


4. 也可以启动Nginx或Apache等Web服务器，编写html页面进行测试（也可以直接打开）





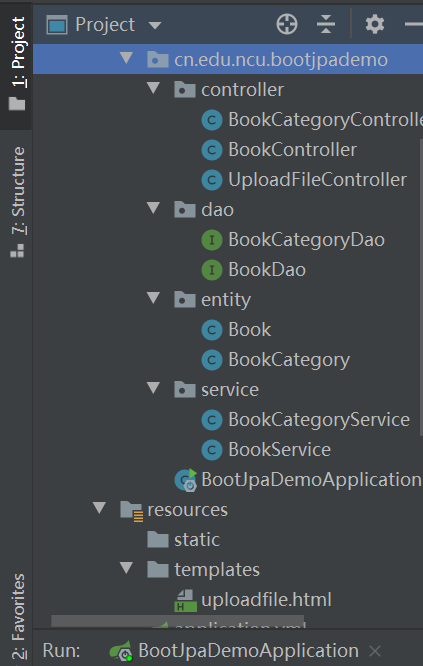
打开文件夹，可以看到文件成功上传了！



六、实验数据及处理结果

每个步骤的代码和运行结果参上。

整个项目的目录结构如下所示：



七、思考讨论题或体会或对改进实验的建议

1. JPA如何将post中的数据存入数据库的？

简单的CRUD功能JPA已经提供，不需要手工完成dao的实现类，只需要调用save方法即可实现更新。

对于实体类 BookCategory，调用 save(bookCategory) 方法，分为以下情况讨论：

（1）假如 bookCategory的主键字段（@Id 修饰的字段）为空，是新增方法，执行一条 insert 语句。

（2）假如主键字段不为空则会先执行 select 语句，查看当前主键对应的 bookCategory是否存在。

（3）假 如不存在则执行 insert 语句，需要注意的是如果主键有自动生成规则，保存的主键则会根据数据库的规则生成而不是设置的值。

（4）假如存在则比较字段值有无变化，有变化则执行 update 语句。

总结如下：

（1）执行 insert 语句的条件：待保存的实体类主键为空，或者主键是一个在数据库中不存在的值。

（2）执行 update 语句的条件：待保存的实体类主键不为空，并且主键是一个在数据库中存在的值，并且相比数据库存在的值有变化。

2. 使用JPA和Mybais的区别

1.ORM映射不同:

　　Mybatis是半自动的ORM框架，提供数据库与结果集的映射；

　　JPA（Hibernate）是全自动的ORM框架，提供对象与数据库的映射；

2.可移植性不同：

　　JPA（Hibernate）通过它强大的映射结构和hql语言，大大降低了对象与数据库(oracle、mysql等)的耦合性

　　Mybatis由于需要手写sql，因此与数据库的耦合性直接取决于程序员写sql的方法，如果sql不具通用性而用了很多某数据库特性的sql语句的话，移植性也会随之降低很多，成本很高。

3.日志系统的完整性不同:

　　JPA（Hibernate）日志系统非常健全，涉及广泛，包括：sql记录、关系异常、优化警告、缓存提示、脏数据警告等;而Mybatis则除了基本记录功能外，功能薄弱很多。

八、参考资料

1.JavaWeb程序设计