# IDEA中开发简单Spring Boot微服务项目

开发一个简单微服务项目，包含两个微服务：Rest API微服务（用户微服务：myshop-user），Web UI微服务（购票微服务：myshop-web）。Rest API微服务连接数据库，Web UI微服务通过调用Rest API微服务获取数据库中数据。

服务名、类名（除部分特殊类）及数据库中数据请自定义，避免与课件雷同！

1. 搭建Rest API微服务
2. 建立Maven项目
3. 建父工程myshop-parent，pom.xml导入以下依赖

<!-- SpringBoot项目必须导入的parent依赖-->

<parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.0.1.RELEASE</version>  
 <relativePath/> <!-- lookup parent from repository -->  
</parent>  
<!-- 定义常用的属性-->

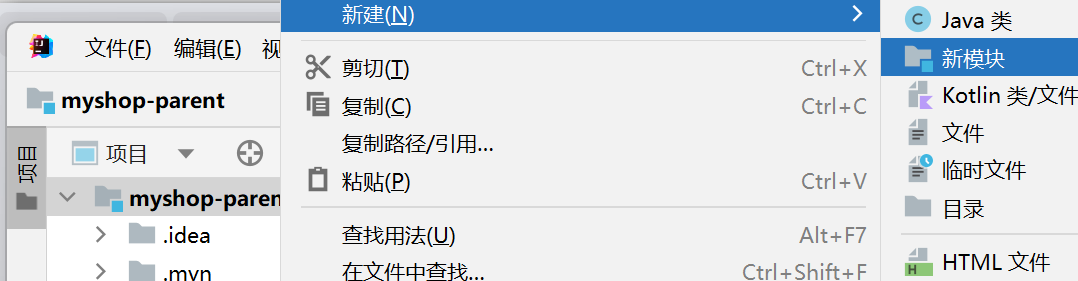
<properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
 <java.version>1.8</java.version>  
</properties>  
<dependencies>

<!-- SpringBoot 对web支持， SpringMVC相关功能，json转换的功能等等-->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
</dependencies>

<!-- maven构建-->

<build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
</build>

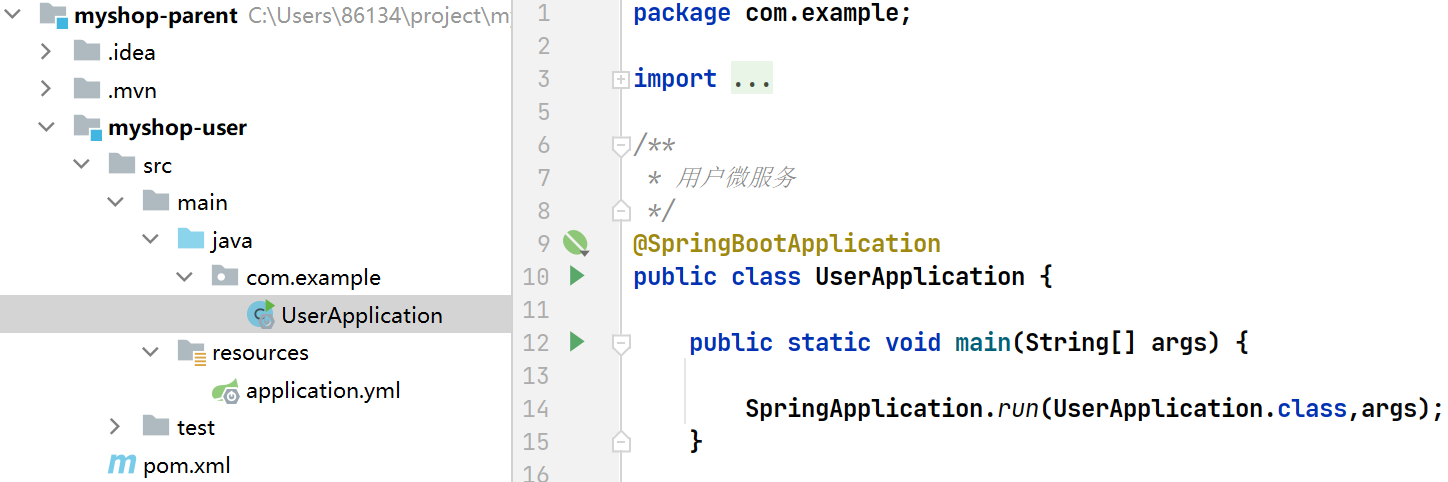
1. 建子工程myshop-user（Rest API用户微服务名）



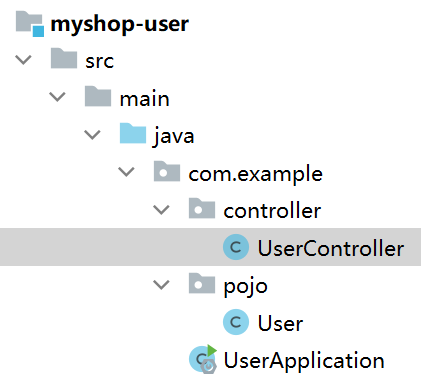
1. 编写配置文件application.yml



1. 编写启动类UserApplication



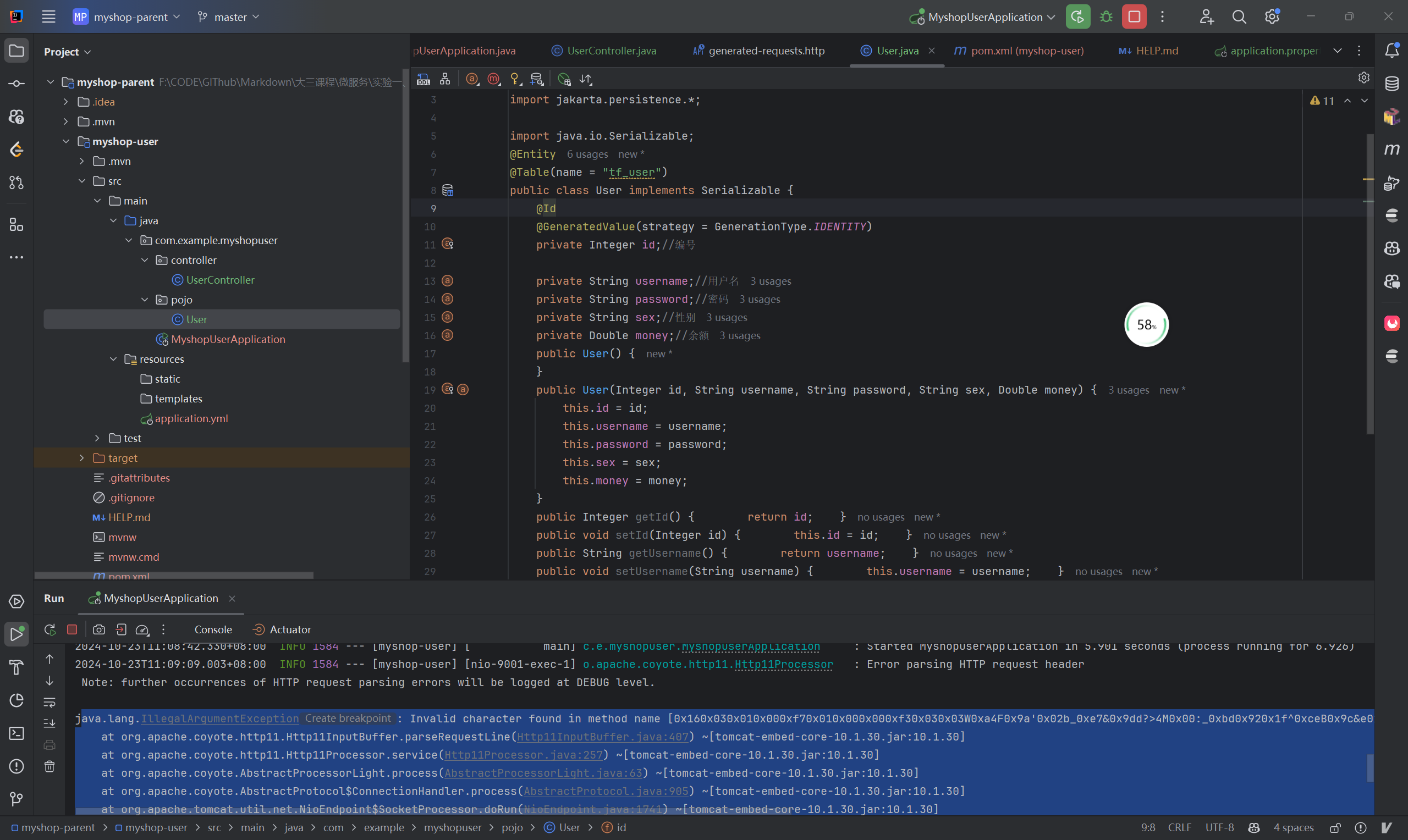
1. 编写UserController（controller包中写UserController类）



先使用模拟用户数据测试微服务，后续将连接到数据库

1. 编写User实体类（pojo包中User类）

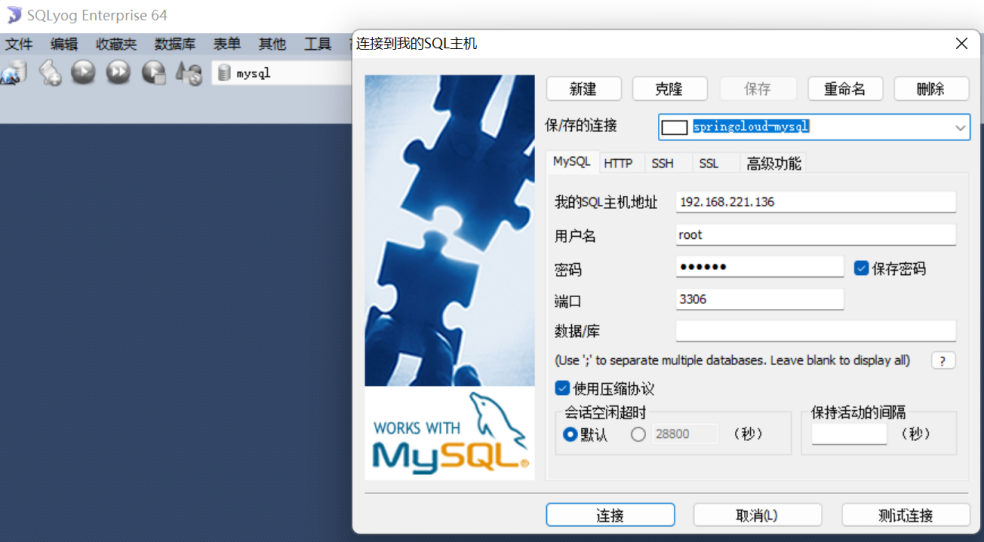
/\*\*  
 \* 用户实体  
 \*/  
  
public class User implements Serializable {  
 private Integer id;//编号  
 private String username;//用户名  
 private String password;//密码  
 private String sex;//性别  
 private Double money;//余额  
 public User() {  
 }  
 public User(Integer id, String username, String password, String sex, Double money) {  
 this.id = id;  
 this.username = username;  
 this.password = password;  
 this.sex = sex;  
 this.money = money;  
 }  
 public Integer getId() { return id; }  
 public void setId(Integer id) { this.id = id; }  
 public String getUsername() { return username; }  
 public void setUsername(String username) { this.username = username; }  
 public String getPassword() { return password; }  
 public void setPassword(String password) { this.password = password; }  
 public String getSex() { return sex; }  
 public void setSex(String sex) { this.sex = sex; }  
 public Double getMoney() { return money; }  
 public void setMoney(Double money) { this.money = money; }  
}



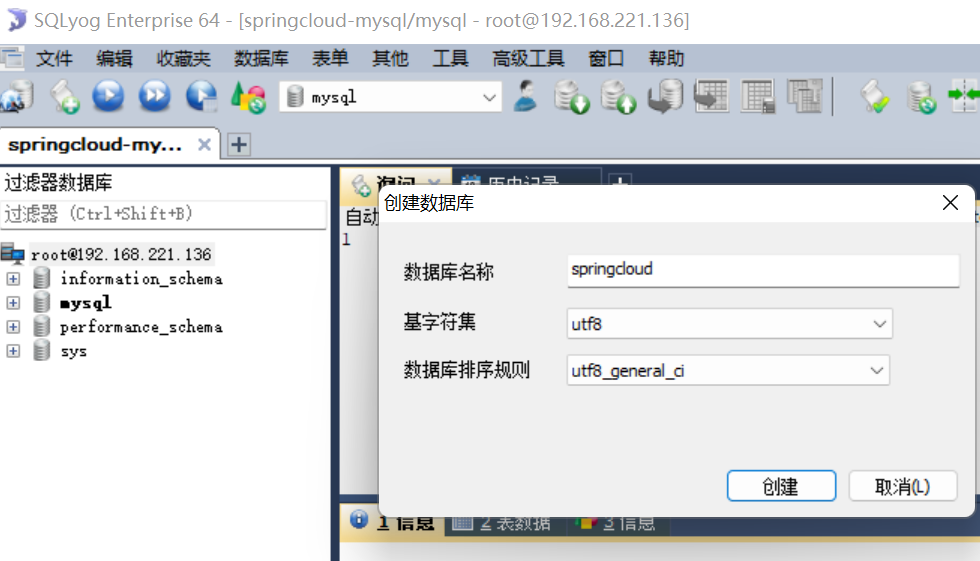
1. 启动并Postman测试，记录测试结果



1. Rest API微服务连接数据库
2. 按lab1中第四步步骤启动Mysql
3. SQLyog连接虚机中Mysql



1. 创建新数据库，本例数据库名为springcloud



1. 整合Spring Data JPA-完成用户CRUD

JPA：java的持久层规范接口，实现对象到数据库表的关系映射

1. 在myshop-user中导入spring data jpa依赖

<dependencies>  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>

</dependencies>

1. 配置application.yml数据源

server:

port: 9101

spring: # 服务名称

application:

name: myshop-user

datasource:

url: jdbc:mysql://192.168.221.136:3306/springcloud?characterEncoding=UTF8&useSSL=false

driver-class-name: com.mysql.jdbc.Driver

username: root

password: 123456

jpa:

show-sql: true # 是否打印sql语句

generate-ddl: true # 是否自动建表

database: mysql # jpa连接的数据库类型

1. Pojo实体映射

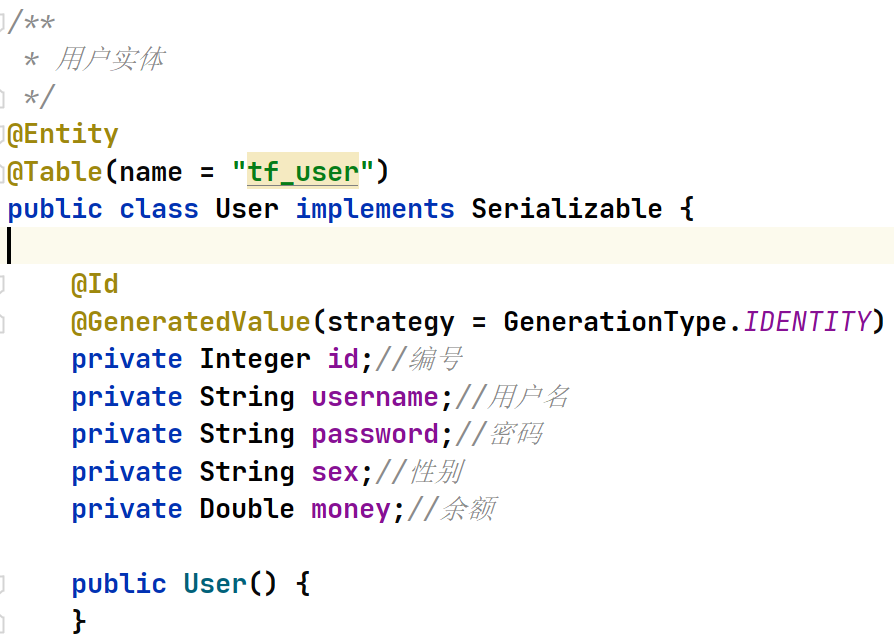
实体类User加以下注解：

@Entity， jpa映射

@Table，将在数据库中生成指定名称的数据表，这里写一个表名

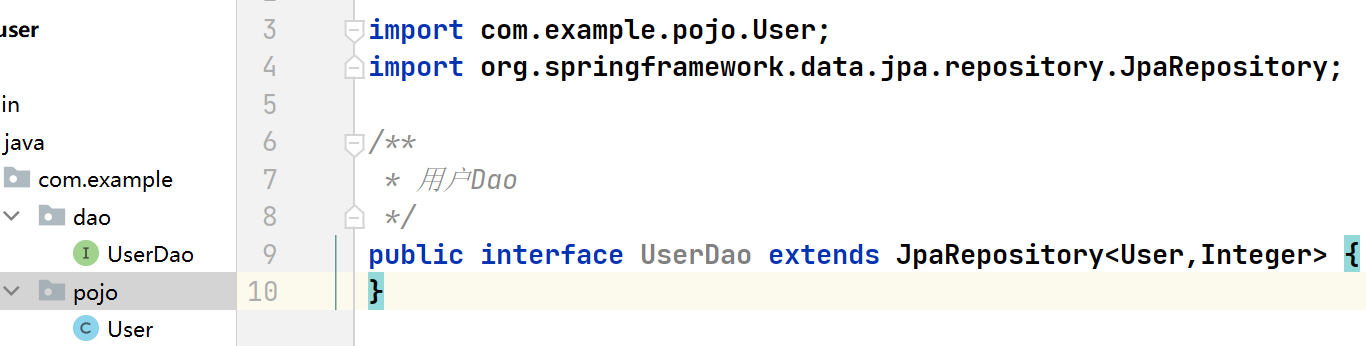
@Id ，实体对象必须有一个唯一标识

@GenerateValue，id自动生成策略，这里用了IDENTITY，自增长。

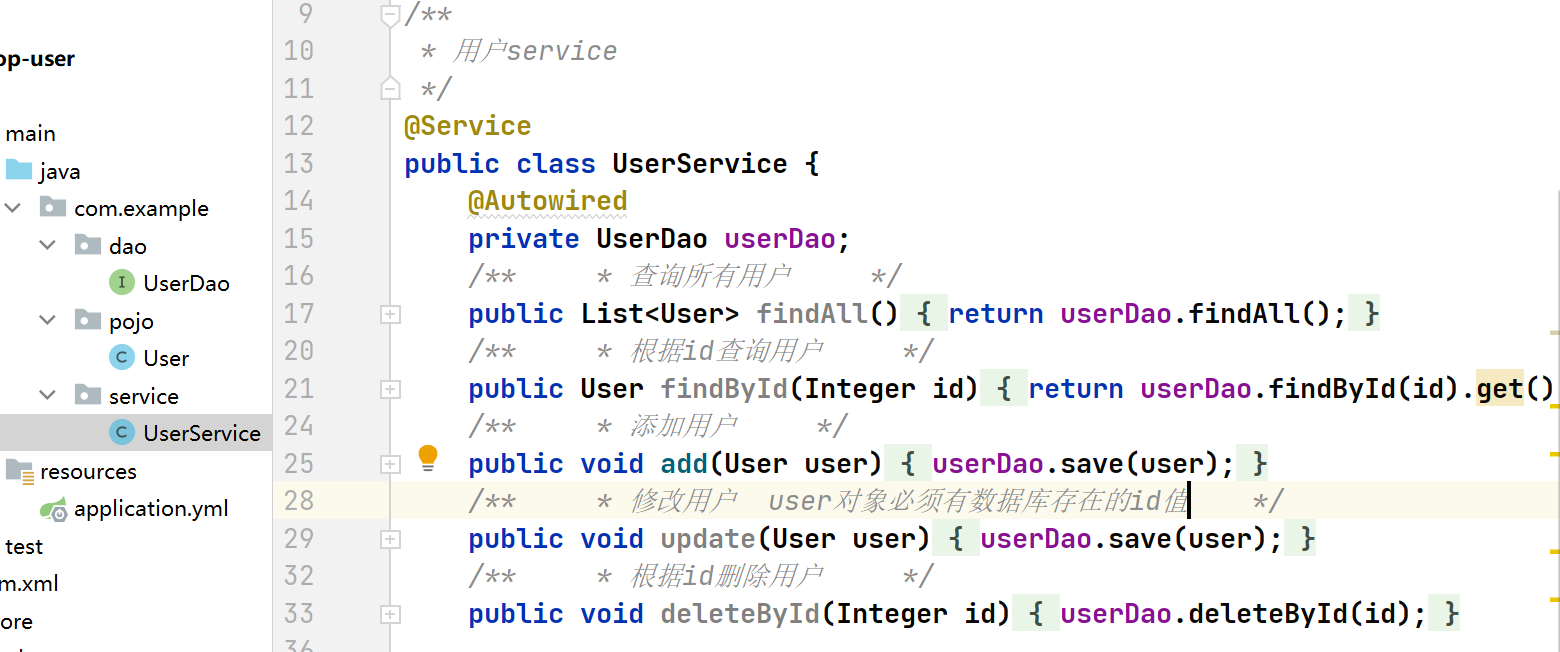


1. 为实体创建jpa存储库接口

dao包中创建UserDao接口



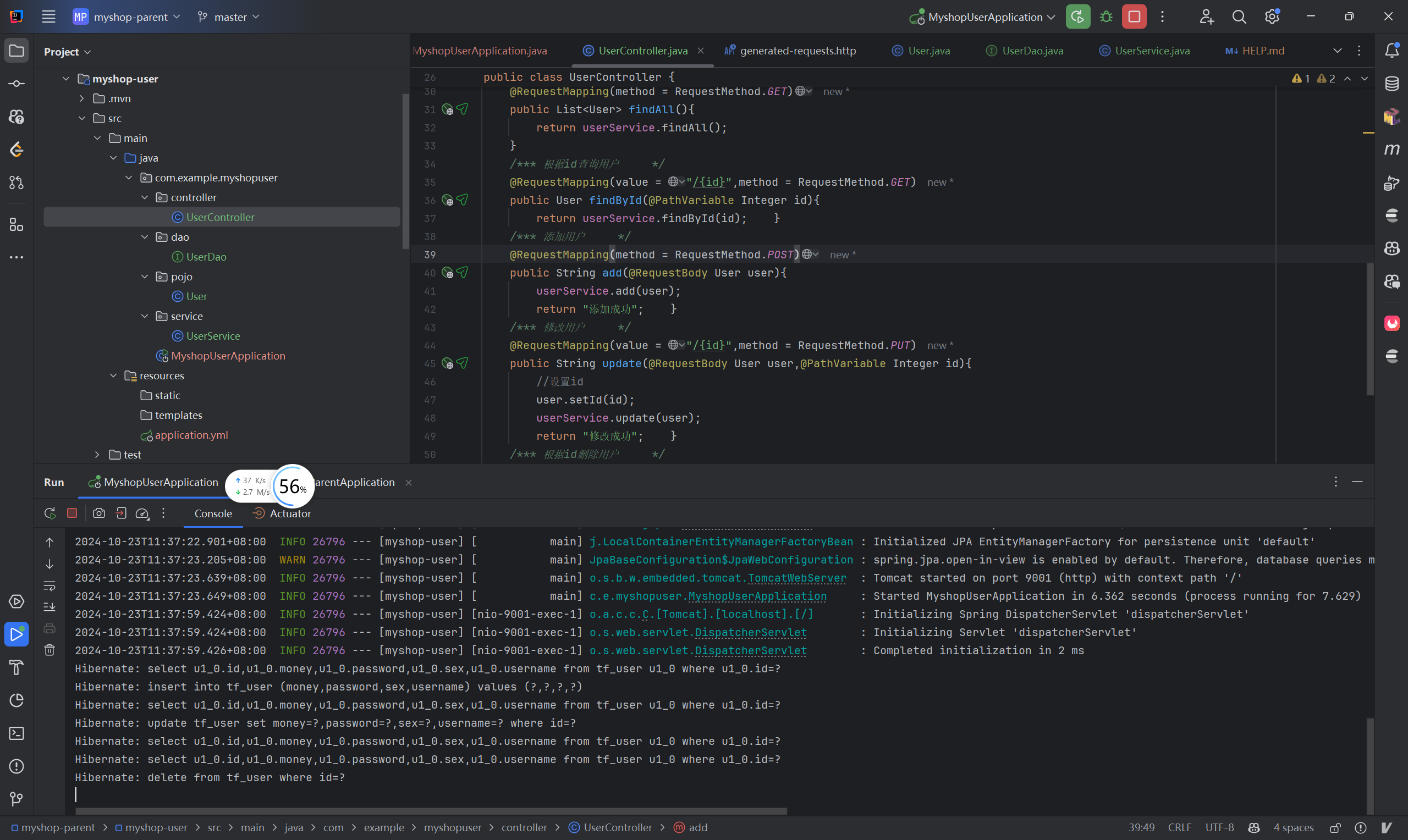
1. 编写Service



/\*\*  
 \* 用户service  
 \*/  
@Service  
public class UserService {  
 @Autowired  
 private UserDao userDao;  
 /\*\* \* 查询所有用户 \*/  
 public List<User> findAll(){ return userDao.findAll(); }  
 /\*\* \* 根据id查询用户 \*/  
 public User findById(Integer id){ return userDao.findById(id).get(); }  
 /\*\* \* 添加用户 \*/  
 public void add(User user){ userDao.save(user); }  
 /\*\* \* 修改用户 up \*/  
 public void update(User user){ userDao.save(user); }  
 /\*\* \* 根据id删除用户 \*/  
 public void deleteById(Integer id){ userDao.deleteById(id); }  
}

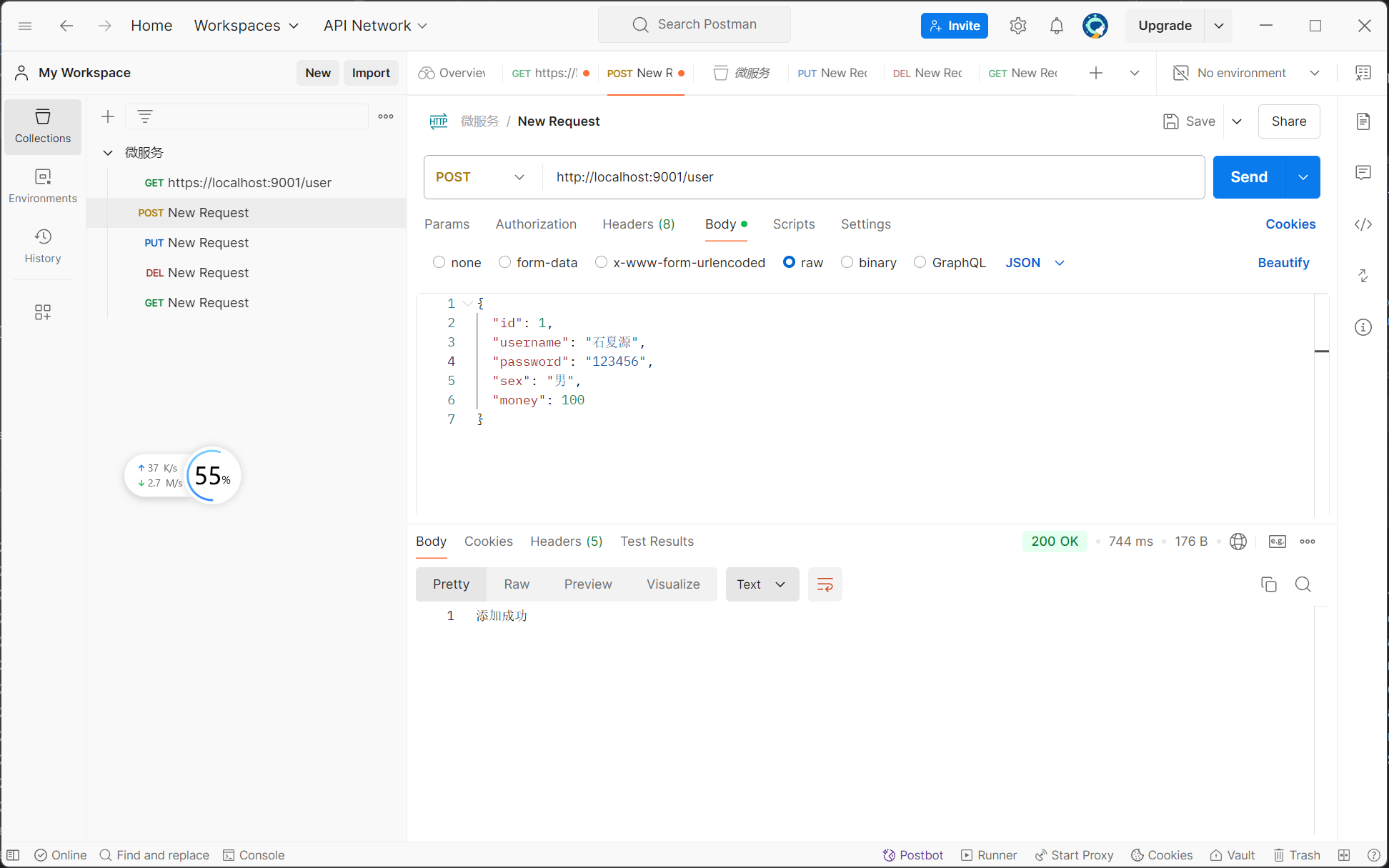
1. 编写Controller

/\*\*\* 用户Controller \*/  
@RequestMapping("/user")  
@RestController // @RestController=@RequestMapping + @ResponseBody  
//@Api(description = "用户控制器")  
public class UserController {  
 @Autowired  
 private UserService userService;  
 /\*\*\* 查询所有用户 \*/  
 @RequestMapping(method = RequestMethod.GET)  
 public List<User> findAll(){  
 return userService.findAll();  
 }  
 /\*\*\* 根据id查询用户 \*/  
 @RequestMapping(value = "/{id}",method = RequestMethod.GET)  
 public User findById(@PathVariable Integer id){  
 return userService.findById(id); }  
 /\*\*\* 添加用户 \*/  
 @RequestMapping(method = RequestMethod.POST)  
 public String add(@RequestBody User user){  
 userService.add(user);  
 return "添加成功"; }  
 /\*\*\* 修改用户 \*/  
 @RequestMapping(value = "/{id}",method = RequestMethod.PUT)  
 public String update(@RequestBody User user,@PathVariable Integer id){  
 //设置id  
 user.setId(id);  
 userService.update(user);  
 return "修改成功"; }  
 /\*\*\* 根据id删除用户 \*/  
 @RequestMapping(value = "/{id}",method = RequestMethod.DELETE)  
 public String deleteById(@PathVariable Integer id){  
 userService.deleteById(id);  
 return "删除成功"; }  
}

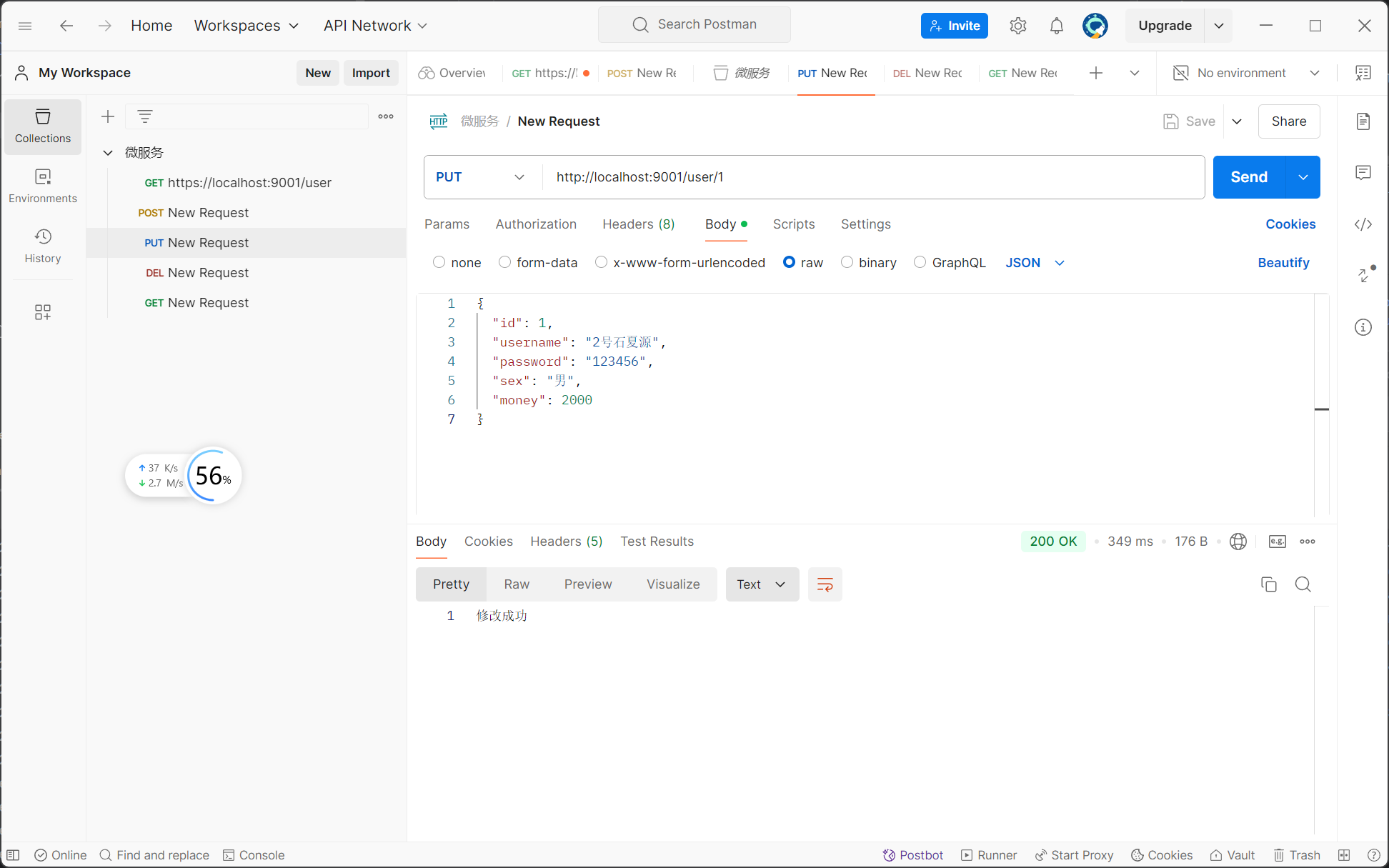


1. Postman测试数据库表增删改查功能，并记录结果。

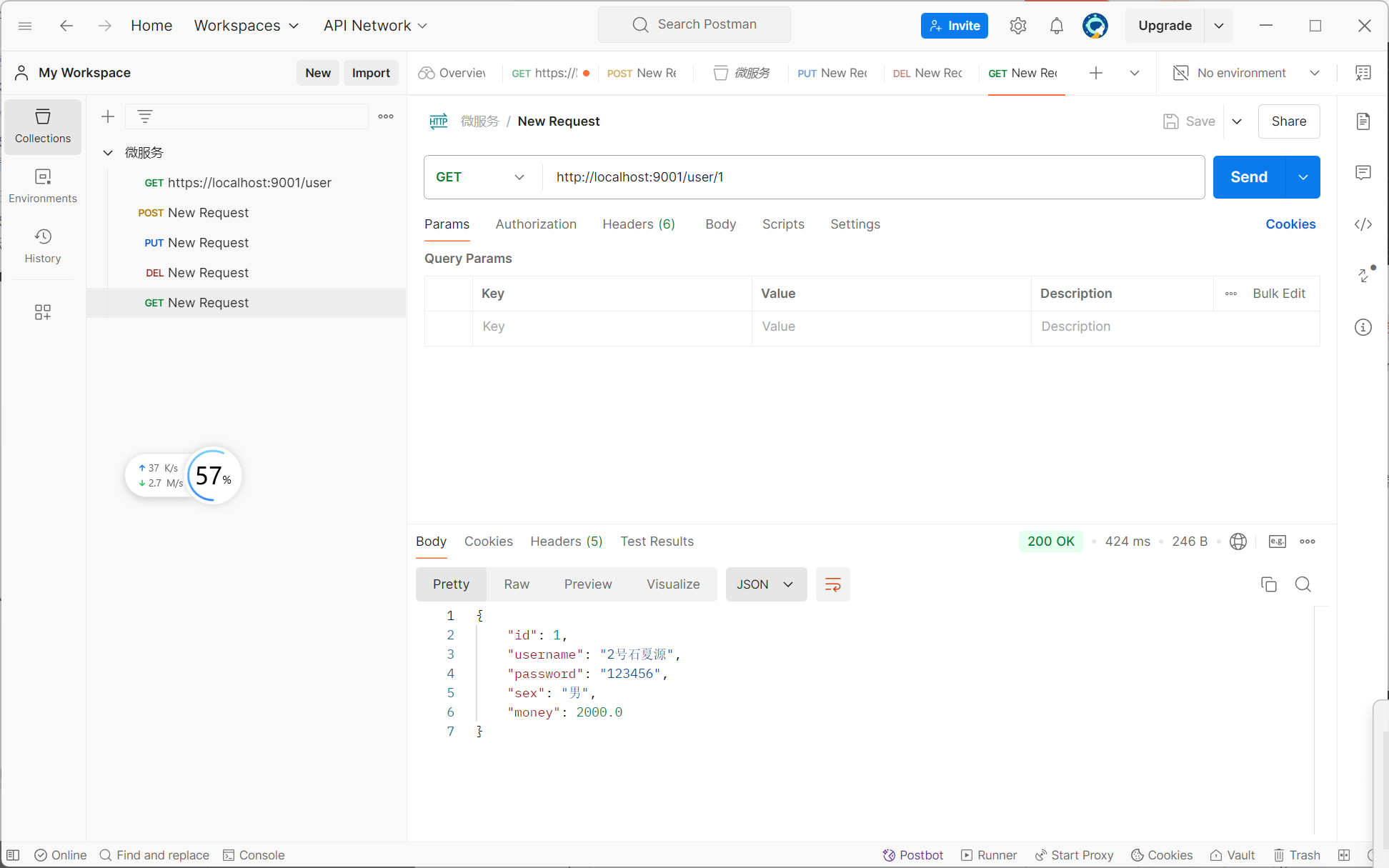
增加：



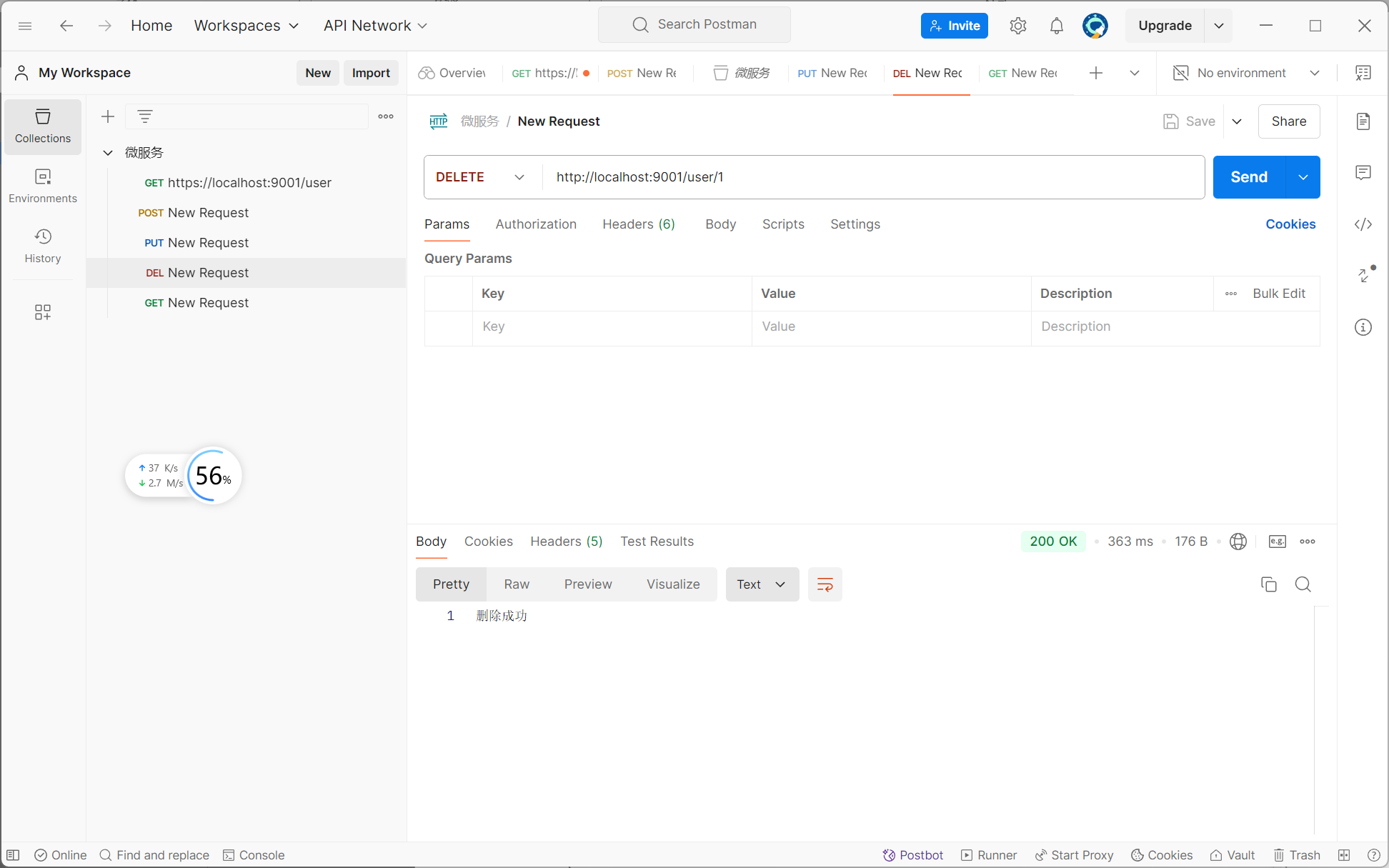
修改：



查：

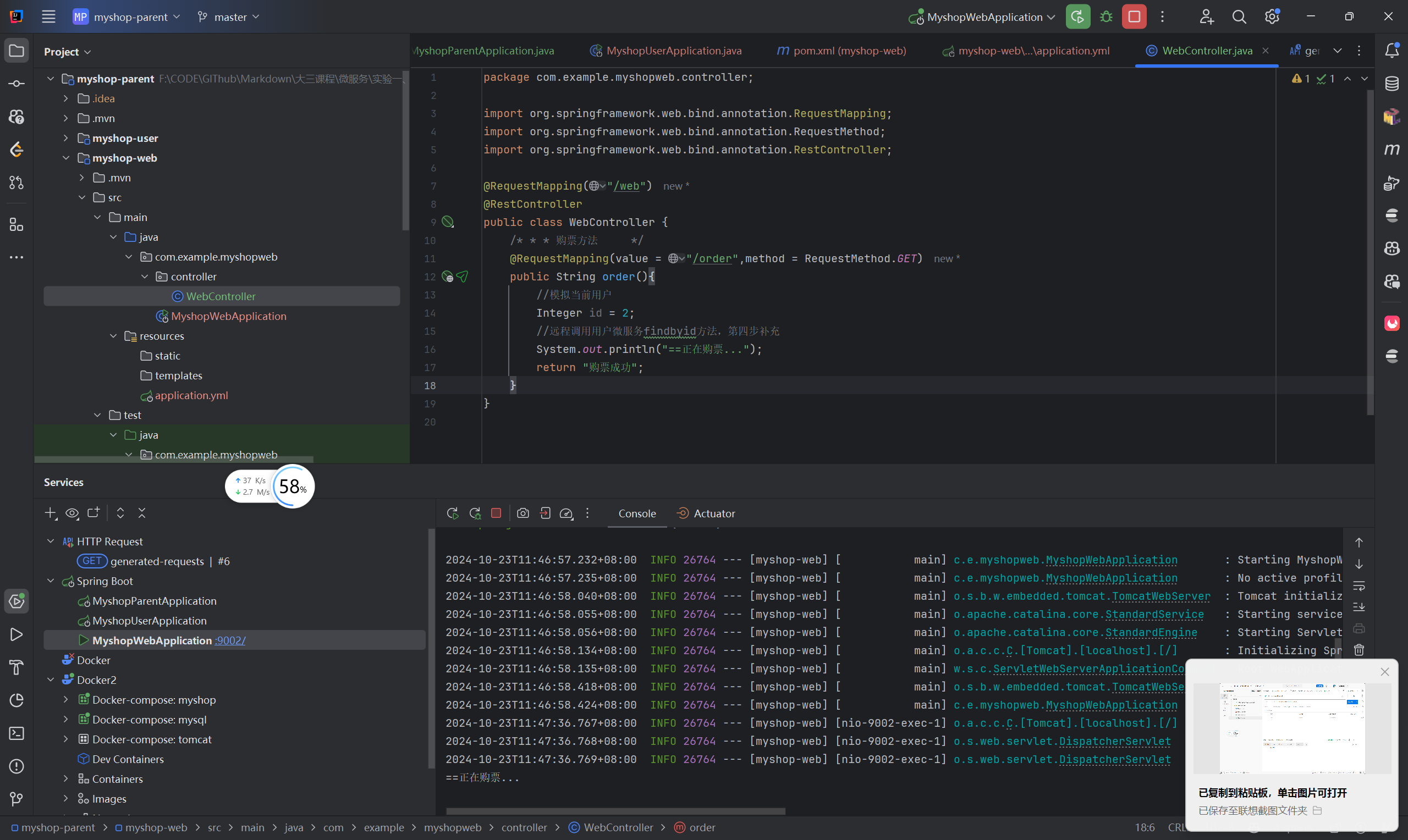


删：

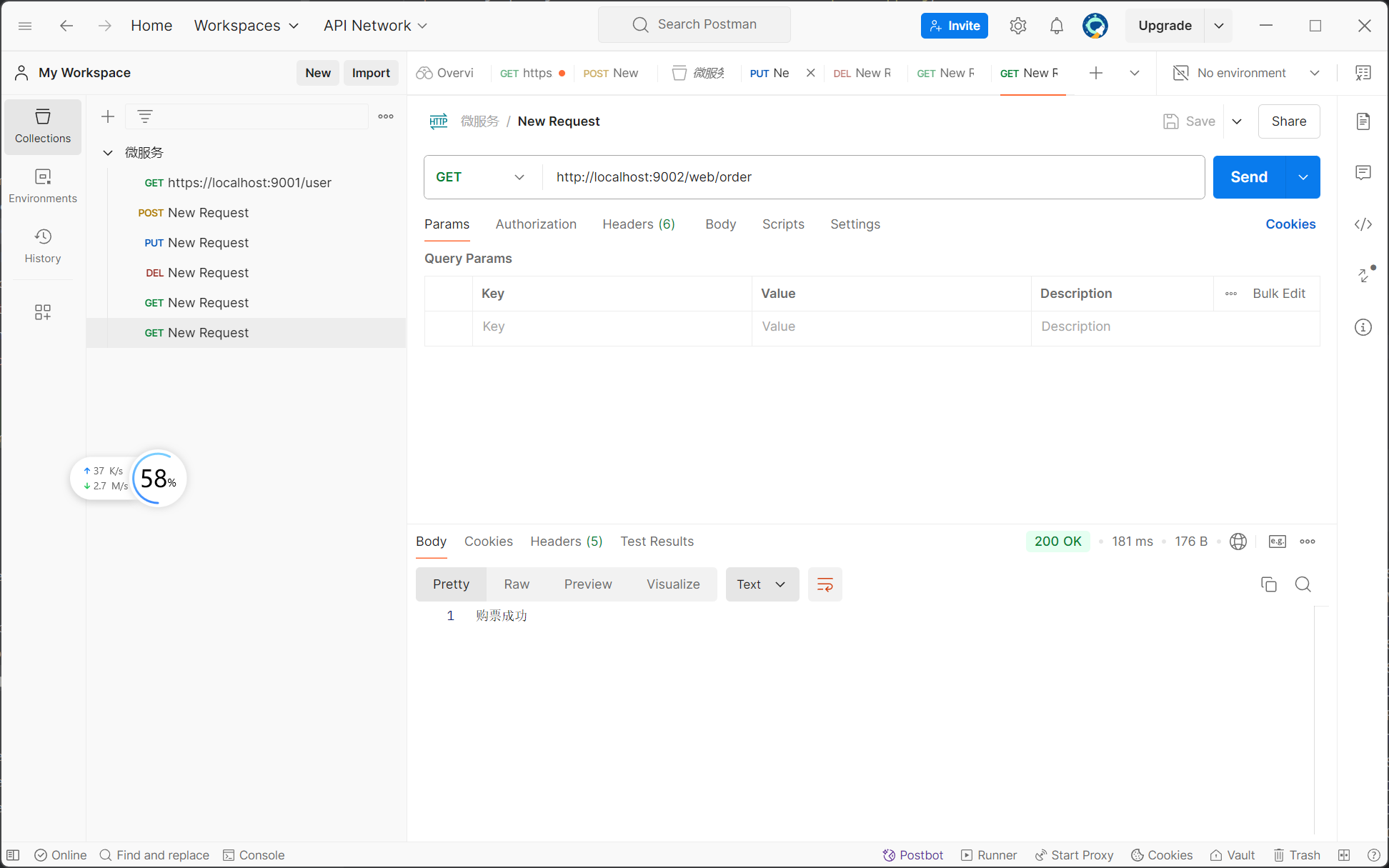


1. 搭建Web UI微服务
2. 建子工程myshop-web（Web UI购票微服务名）
3. 编写配置文件application.yml，port为9002
4. 编写启动类UserApplication
5. 编写WebController

/\*\*\* 购票Controller \*/  
@RequestMapping("/web")  
@RestController  
public class WebController {  
 /\* \* \* 购票方法 \*/  
 @RequestMapping(value = "/order",method = RequestMethod.GET)  
 public String order(){  
 //模拟当前用户  
 Integer id = 2;  
 //远程调用用户微服务findbyid方法，第四步补充  
 System.out.println("==正在购票...");  
 return "购票成功";  
 }  
}



1. Postman测试Web UI微服务order方法是否可被调用



1. RestTemplate实现远程调用
2. 在购票微服务启动类初始化RestTemplate



1. 修改购票方法order（），使用RestTemplae远程调用用户微服务方法



1. 重启微服务，postman测试请求order方法，记录请求返回结果以及IDEA控制台的两个微服务信息。

