# Jedis客户端代码

# 1.环境

## 1.1启动redis三个实例

文本

描述已自动生成

## 1.2 开启6379、6380、6381端口号



# 2 创建测试工程（quickster工程）

## 2.1创建spring boot工程（quickster工程）

**springboot-redis-test01**

图形用户界面, 文本, 应用程序

中度可信度描述已自动生成

## 2.2 pom文件

1. 继承spring-boot-starter-parent

继承easymall-parent,使其成为标准的springboot项目

图表, 条形图

描述已自动生成

1. 添加1.4.7.RELEASE版本的redis依赖支持

<!-- 添加支持redis的简化依赖 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-redis</artifactId>

<version>1.4.7.RELEASE</version>

</dependency>

# Jedis连接操作redis

Jedis中的方法名称和redis的命令名称几乎一样

## 3.1.jedis的连接

文本

描述已自动生成

文本

描述已自动生成

## 存取hash

文本

描述已自动生成

文本

描述已自动生成

## 存取List

文本

描述已自动生成

文本

描述已自动生成

## 存取Set

图形用户界面, 文本

描述已自动生成

图片包含 文本

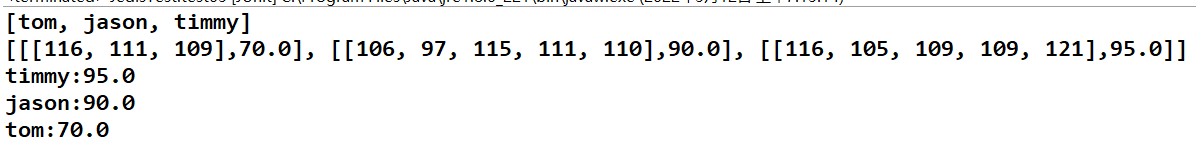
描述已自动生成

## 存取ZSet实现排序

**存储zset类型数据，取出zset数据并进行解析**

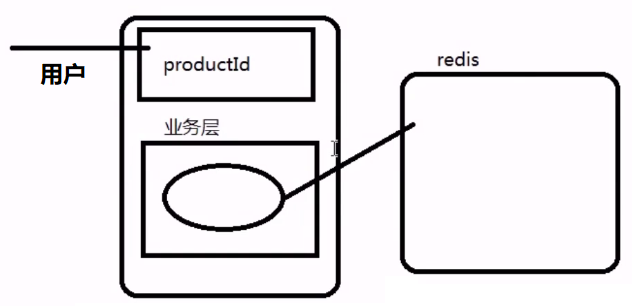
文本

描述已自动生成



# 模拟缓存的逻辑(String)

查询一个商品的缓存数据



用户查询一个商品 productId=05e20c1a-0401-4c0a-82ab-6fb0f37db397

逻辑:

1）判断缓存是否存在这个商品数据

2）如果存在,读取出来.直接封装返回,不再调用持久层

3）如果不存在,调用持久层获取数据,并存储一份在redis供后续使用。

4）存储数据时,key值的设计规则:根据业务意义设计key

例如,某个商品的key="**product\_**"+productId;

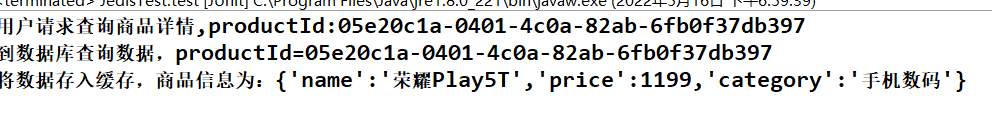
某个用户的key="user\_"+userId;

代码：

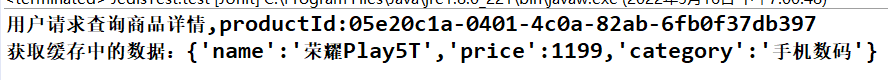
文本

描述已自动生成

第一次执行结果：



第二次执行结果：



作业:将缓存逻辑迁移到product商品中心的单个商品查询功能中

提示：

1. 将返回值Product类型改成String类型

文本

描述已自动生成

2）将Product对象转换成json

import com.fasterxml.jackson.databind.ObjectMapper;

// jackson代码.将product对象转化成json

ObjectMapper mapper = **new** ObjectMapper();

String productJson;

**try** {

productJson = mapper.writeValueAsString(product);

} **catch** (Exception e) {

e.printStackTrace();

**return** "";

}

// 缓存到Redis

………………