配置

Redission提供一些高级服务：远程调用\*分布式锁\*分布式对象、容器

SpringBoot集成Redission，引入以下依赖：

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>org.redisson</groupId>

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

<version>3.13.0</version>

</dependency>

</dependencies>

Peoperties中配置：

**# Redis服务器地址**

**spring.redis.host=212.129.234.68**

**# Redis 服务器端口号**

**spring.redis.port=6379**

**# Redis 服务器密码 （请换成自己机器密码）**

**spring.redis.password=Hello122342**

分布式ID

应用实例

**@Controller**

**public** **class** **AutoIdController** {

**@Autowired**

RedissonClient redissonClient;

**@GetMapping("/getautoid")**

**@ResponseBody**

**public** String **getAutoId**() {

*//格式化格式为年月日*

DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("yyyyMMdd");

*//获取当前时间*

String now = LocalDate.now().format(dateTimeFormatter);

*//通过redis的自增获取序号*

RAtomicLong atomicLong = redissonClient.getAtomicLong(now);

atomicLong.expire(1, TimeUnit.DAYS);

*//拼装订单号*

**return** now + "" + atomicLong.incrementAndGet();

}

}

分布式锁

Synchronized本地锁——只能解决一台服务器并发问题

Redis分布式锁——锁所有服务器的锁

Redis发布锁三步：

1.取得锁

//CUSTOM\_NAME 自定义锁名称字符串，一般是跟业务相关的名称

//Rlock 继承于 java.util.concurrent.locks.Lock;

RLock rLock = redissonClient.getLock("CUSTOM\_NAME")

2.上锁

/\*\*

\* 上锁过程

\* 两种常用上锁方式tryLock()或者lock()

\*/

rLock.tryLock();

3.解锁

//解锁

rLock.unlock();

RLock transferLock = redissonClient.getLock(“PURCHASE”);

transferLock.lock();

try{

商品购买实现代码断

}catch(Exception e){

logger.error(“”,e);

//private static final Logger logger = LoggerFactory.getLogger(“xxxx.class”);

}finally{

transferLock.unlock();//无论是否出现异常，一定解锁

}

事务：撤销并发操作

分布式锁：阻止并发操作