使用Redis优化“地址暴露接口”

1. redis的安装使用
2. 在官网上下载redis，进行解压
3. 打开一个cmd窗口，运行redis-server.exe（如果没有配置环境变量则需要去redis目录下执行该命令）
4. 另外启动一个cmd窗口，原来的不要关闭，不然就无法访问服务端了
5. 运行 redis-cli.exe –h 127.0.0.1 –p 6379
6. 用java访问搭建好的本地redis

添加redis java客户端依赖

|  |
| --- |
| <!-- redis客户端：Jedis -->  <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  <version>2.7.3</version>  </dependency> |

新建一个redisDao

|  |
| --- |
| **public** **class** RedisDao {    **private** Logger logger = LoggerFactory.*getLogger*(**this**.getClass());  **private** JedisPool jedisPool; //相当于数据库的连接池    **public** RedisDao(String ip,**int** port){  jedisPool = **new** JedisPool(ip,port);  }    **private** RuntimeSchema<Seckill> schema = RuntimeSchema.*createFrom*(Seckill.**class**);    **public** Seckill getSeckill(**long** seckillId){  //redis操作逻辑  **try** {  Jedis jedis = jedisPool.getResource(); //相当于数据库的connection  **try** {  String key = "seckill:"+seckillId;  //并没有实现内部序列化操作  //get->byte[] -> 反序列化 -> Object(Seckill)  //采用自定义序列化  //prootostuff:pojo(必须是有get(),set()标准的java对象)  **byte**[] bytes = jedis.get(key.getBytes());  //缓存获取到  **if**(bytes != **null**){  //空对象  Seckill seckill = schema.newMessage();  ProtostuffIOUtil.*mergeFrom*(bytes, seckill, schema);  //seckill 被反序列化  **return** seckill;  }  } **finally** {  jedis.close();  }  }  **catch** (Exception e) {  logger.error(e.getMessage(),e);  }  **return** **null**;  }    **public** String putSeckill(Seckill seckill){  //set Object(Seckill) -> 序列化 -> byte[]  **try** {  Jedis jedis = jedisPool.getResource();  **try** {  String key = "seckill:"+seckill.getSeckillId();  **byte**[] bytes = ProtostuffIOUtil.*toByteArray*(seckill, schema,  LinkedBuffer.*allocate*(LinkedBuffer.***DEFAULT\_BUFFER\_SIZE***));  //超时缓存  **int** timeout = 60 \* 60;//1小时  String result = jedis.setex(key.getBytes(), timeout, bytes);  **return** result;  } **finally** {  jedis.close();  }  } **catch** (Exception e) {  logger.error(e.getMessage(),e);  }  **return** **null**;  }  } |

添加依赖

|  |
| --- |
| <!-- RedisDao -->  <bean id=*"redisDao"* class=*"edu.hubu.seckill.dao.cache.RedisDao"*>  <constructor-arg index=*"0"* value=*"127.0.0.1"*></constructor-arg>  <constructor-arg index=*"1"* value=*"6379"*></constructor-arg>  </bean> |

在service中实现优化

|  |
| --- |
| //优化点:缓存优化:超时的基础上维护一致性  /\*\*  \* get from cache  \* if null  \* get db  \* else  \* put cache  \* logic  \*/  //1. 访问dedis  Seckill seckill = redisDao.getSeckill(seckillId);  **if**(seckill == **null**){  //2.访问数据库  seckill = seckillDao.queryById(seckillId);  **if**(seckill != **null**){  //3.加入redis  redisDao.putSeckill(seckill);  }**else**{  //如果秒杀不存在  **return** **new** Exposer(**false**, seckillId);  }  } |