**Redis个人理解和总结**

redis 是Nosql数据库， key-value存储系统。 基于内存运行，性能高效。

低延迟的读写速度。应用广泛：可以做缓存系统（缓存热点数据），计数器，排行榜等。

**有五种自有数据类型：string,hash,list,set,zset.**

string 类型 二进制安全的字符串，可以存储字符串，图片，视频等，最大长度512M，做一些复杂的计数功能的缓存。

hash类型，存放的是结构化对象，方便于操作其中的字段，可以用在单点登录上，将cookieId作为key,用户信息存放在value中。

list：插入顺序排序的字符串元素集合，基于双链表。

set:无顺序集合，元素唯一,底层hash表，使用交并差集，可以做共同爱好，个性化喜好等功能。

zset带权重参数的集合 底层hash表，可以做排行榜等功能。

**redis的持久化机制：**

持久化机制有两种，1.RDB(redis默认使用)，通过时间周期（通过配置文件中的save参数定义快照周期）把内存中的数据以快照的形式保存到磁盘中，产生dump.rdb的数据文件。

1. AOF,redis会将每个收到的写命令都通过write函数追加到文件最后。同时开启两种机制，数据恢复redis会优先选择AOF恢复。

**Redis的缓存击穿，缓存雪崩，缓存预热：**

**缓存击穿**：是指用户查询的数据在数据库中没有，并且大量查询该数据，导致缓存中不存在该数据，数据库中也没有，增加了很多无效的查询，增大了数据库的访问压力，特别是一些恶意攻击。

解决该问题的方法是采用布隆过滤器，它的核心思想是利用多个hash函数完成对数据的判重，当数据通过所有hash函数判断数据存在于集合，则存在，没有则直接返回空数据。

**缓存雪崩**：缓存雪崩是指某个时间，大量缓存失效了，大量的访问直接查询数据库，导致数据库压力很大，甚至超出数据库的承载压力。应对方案可以是：采用多redis的方式，多redis之间设置有间隔的数据过期时间。

**缓存预热**:对于一部分热点数据，在服务开启时将热点数据加载进缓存中，例如秒杀活动，可以在活动开始前五分钟，将活动信息都加载进缓存中，可以是定时刷新缓存。

**Redis的过期策略以及内存淘汰机制：**

**Redis采用定期删除+惰性删除策略**，定期删除是指，默认在每100ms随机抽查是否有过期的key,如果有则删除。惰性删除是指，当用户获取某个key时，redis会检查是否过期，过期了则删除。

1. **内存淘汰策略有**：（在redis.conf中有一行配置 maxmemory-policy）  
   volatile-lru：从已设置过期时间的数据集（server.db[i].expires）中挑选最近最少使用的数据淘汰

volatile-ttl：从已设置过期时间的数据集（server.db[i].expires）中挑选将要过期的数据淘汰

volatile-random：从已设置过期时间的数据集（server.db[i].expires）中任意选择数据淘汰

allkeys-lru：从数据集（server.db[i].dict）中挑选最近最少使用的数据淘汰

allkeys-random：从数据集（server.db[i].dict）中任意选择数据淘汰。

**redis实现分布式锁**

Setnx lock-key value1

Setnx lock-key value2

Get lock-key

**redis在springboot中的使用：**

导入依赖

<!--redis-->

<dependency>

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

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

</dependency>

配置连接信息

# Redis数据库索引（默认为0）

spring.redis.database=0

# Redis服务器地址

spring.redis.host=192.168.0.24

# Redis服务器连接端口

spring.redis.port=6379

# Redis服务器连接密码（默认为空）

spring.redis.password=

# 连接池最大连接数（使用负值表示没有限制）

spring.redis.pool.max-active=200

# 连接池最大阻塞等待时间（使用负值表示没有限制）

spring.redis.pool.max-wait=-1

# 连接池中的最大空闲连接

spring.redis.pool.max-idle=10

# 连接池中的最小空闲连接

spring.redis.pool.min-idle=0

# 连接超时时间（毫秒）

spring.redis.timeout=1000

使用redisTempate

public class Test\_1{

@Autowired

private RedisTemplate<String,String>redisTemplate;

@Test

public void set(){

redisTemplate.opsForValue().set("myKey","myValue");

System.out.println(redisTemplate.opsForValue().get("myKey"));

}

}

创建一个 RedisTemplate<String,Object>的类

@Configuration

public class RedisConfig {

@Bean

@SuppressWarnings("all")

public RedisTemplate<String, Object> redisTemplate(RedisConnectionFactory factory) {

RedisTemplate<String, Object> template = new RedisTemplate<String, Object>();

template.setConnectionFactory(factory);

Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = new Jackson2JsonRedisSerializer(Object.class);

ObjectMapper om = new ObjectMapper();

om.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY);

om.enableDefaultTyping(ObjectMapper.DefaultTyping.NON\_FINAL);

jackson2JsonRedisSerializer.setObjectMapper(om);

StringRedisSerializer stringRedisSerializer = new StringRedisSerializer();

// key采用String的序列化方式

template.setKeySerializer(stringRedisSerializer);

// hash的key也采用String的序列化方式

template.setHashKeySerializer(stringRedisSerializer);

// value序列化方式采用jackson

template.setValueSerializer(jackson2JsonRedisSerializer);

// hash的value序列化方式采用jackson

template.setHashValueSerializer(jackson2JsonRedisSerializer);

template.afterPropertiesSet();

return template;

}

}

操作redis的五种数据类型：

opsForValue()：操作字符串。

opsForList()：操作列表。

opsForHash()：操作哈希。

opsForSet()：操作集合。

opsForZSet()：操作有序集合。

string:

redisTemplate.opsForValue().set("name","linqz",3, TimeUnit.SECONDS);

获取旧值：getAngSet 追加字符串：.append。

list: 双向列表：leftPushAll 左侧插入，rightPushAll右侧插入

redisTemplate.opsForList().rightPush("userInfo",1);

redisTemplate.opsForList().index("userInfo",0)；

redisTemplate.opsForList().rightPushAll("user1",user1);

redisTemplate.opsForList().range("user1",0,-1)；

查询列表长度

redisTemplate.opsForList().size("user1")；

删除列表中的元素

redisTemplate.opsForList().remove("user1",1,"linqz");

弹出元素

redisTemplate.opsForList().rightPop("user1")；

hash

在单点登录的时候存储用户信息，以cookie作为key,设置缓存时间30min.

redisTemplate.opsForHash().putAll("userHash",userMap);

redisTemplate.opsForHash().put("userHash","userName","linqz");

获取特定key的value。

redisTemplate.opsForHash().get("userHash","userName");

获取特定hash的所有value值

redisTemplate.opsForHash().values("userHash");

获取特定hash的所有key

redisTemplate.opsForHash().keys("userHash");

删除特定hash下的特定key

redisTemplate.opsForHash().delete("userHash","userName");

set集合：不存放重复值，无序。可以做去重，计算共同爱好，独有爱好，共同好友等。

redisTemplate.opsForSet().add("citySet",citys)

redisTemplate.opsForSet().remove("citySet",citys)

交集：

redisTemplate.opsForSet().intersect("citySet1","citySet2")；

并集：

redisTemplate.opsForSet().union("citySet1","citySet2")；

差集：

redisTemplate.opsForSet().difference("citySet1","citySet2")；

sorted set

redisTemplate.opsForZSet().add("zset2", "linqz", 9.6);

redisTemplate.opsForZSet().range("zset2", 0, -1);

redisTemplate.opsForZSet().count("zset2", 0, 8);

redisTemplate.opsForZSet().size("zset2");