## 一、配置redis

<!-- 引入 redis 依赖 -->

<dependency>

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

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

</dependency>

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.data</groupId>

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

</dependency>

# REDIS (RedisProperties)

# Redis数据库索引（默认为0）

spring.redis.database=0

# Redis服务器地址

spring.redis.host=localhost

# 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.jedis.pool.max-active=8

# 连接池最大阻塞等待时间（使用负值表示没有限制）

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

# 连接池中的最大空闲连接

spring.redis.jedis.pool.max-idle=8

# 连接池中的最小空闲连接

spring.redis.jedis.pool.min-idle=0

# 连接超时时间（毫秒）

spring.redis.timeout=5000

****使用StringRedisTemplate****

直接在service实现类中注入StringRedisTemplate。

@Autowired

private StringRedisTemplate redisTemplate;

RedisConfig配置类，相当于配置了RedisTemplate、和StringRedisTemplate

@Configuration

public class RedisConfig{

@Autowired

RedisConnectionFactory redisConnectionFactory;

@Bean

public RedisTemplate<String, Object> functionDomainRedisTemplate() {

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

initDomainRedisTemplate(redisTemplate, redisConnectionFactory);

return redisTemplate;

}

/\*

\*设置redisTemplate序列化策略，否则在使用redisTemplate的时候在redis的客户端查看会出现乱码

\*/

private void initDomainRedisTemplate(RedisTemplate<String, Object> redisTemplate, RedisConnectionFactory factory) {

redisTemplate.setKeySerializer(new StringRedisSerializer());

redisTemplate.setHashKeySerializer(new StringRedisSerializer());

redisTemplate.setHashValueSerializer(new JdkSerializationRedisSerializer());

redisTemplate.setValueSerializer(new JdkSerializationRedisSerializer());

redisTemplate.setConnectionFactory(factory);

}

}

## 使用redis

### 简介

* RedisTemplate：

redisTemplate.opsForList();//操作list

redisTemplate.opsForValue();//操作字符串

redisTemplate.opsForCluster();//集群时使用

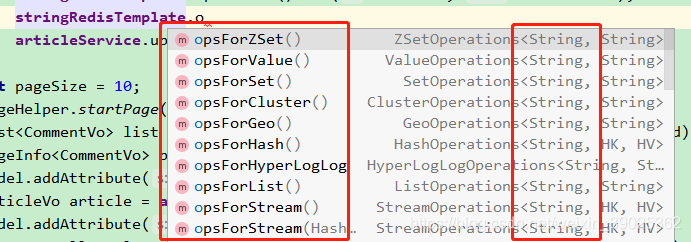
redisTemplate.opsForGeo();//地理位置时使用

redisTemplate.opsForHash();//操作hash

redisTemplate.opsForSet();//操作set

redisTemplate.opsForZSet();//操作有序set

* StringRedisTemplate



### 2、使用

//先引入StringRedisTemplate

@Autowired

private StringRedisTemplate stringRedisTemplate;

123

//向redis里存入数据和设置缓存时间

stringRedisTemplate.opsForValue().set("redis", "100", 60 \* 10, TimeUnit.SECONDS);

//val做-1操作

stringRedisTemplate.boundValueOps("redis").increment(-1);

//根据key获取缓存中的val

stringRedisTemplate.opsForValue().get("redis")

//val +1

stringRedisTemplate.boundValueOps("redis").increment(1);

//根据key获取过期时间

stringRedisTemplate.getExpire("redis");

//根据key获取过期时间并换算成指定单位

stringRedisTemplate.getExpire("redis",TimeUnit.SECONDS);

//根据key删除缓存

stringRedisTemplate.delete("redis");

//检查key是否存在，返回boolean值

stringRedisTemplate.hasKey("redis");

//向指定key中存放set集合

stringRedisTemplate.opsForSet().add("redis", "1","2","3");

//设置过期时间

stringRedisTemplate.expire("redis",1000 , TimeUnit.MILLISECONDS);

//根据key查看集合中是否存在指定数据

stringRedisTemplate.opsForSet().isMember("redis", "1");

//根据key获取set集合

stringRedisTemplate.opsForSet().members("redis");

//验证有效时间

Long expire = stringRedisTemplate.boundHashOps("redis").getExpire();

System.out.println("redis有效时间："+expire+"秒");

### 具体常用的方法的使用

1、add(K key, V value, double score) 添加元素到变量中同时指定元素的分值

redisTemplate.opsForZSet().add("value","A",1);

redisTemplate.opsForZSet().add("value","B",3);

redisTemplate.opsForZSet().add("value","C",2);

2、add(K key, Set<ZSetOperations.TypedTuple> tuples) 添加tuples到排序集key，或者score如果已存在则更新它。

ZSetOperations.TypedTuple<Object> typedTuple1 = new DefaultTypedTuple<Object>("A",6.0);

ZSetOperations.TypedTuple<Object> typedTuple2 = new DefaultTypedTuple<Object>("B",7.0);

ZSetOperations.TypedTuple<Object> typedTuple3 = new DefaultTypedTuple<Object>("C",5.0);

Set<ZSetOperations.TypedTuple<Object>> typedTupleSet = new HashSet<ZSetOperations.TypedTuple<Object>>();

typedTupleSet.add(typedTuple1);

typedTupleSet.add(typedTuple2);

typedTupleSet.add(typedTuple3);

redisTemplate.opsForZSet().add("typedTupleSet",typedTupleSet);

Set valueSet = redisTemplate.opsForZSet().range("typedTupleSet",0,-1);

System.out.println(valueSet);

3、rank(K key, Object o) 获取变量中元素的索引值

long index = redisTemplate.opsForZSet().rank("value","B");

System.out.println(index);

1

2

4、count(K key, double min, double max) 获取区间值的个数。

long count = redisTemplate.opsForZSet().count("zSetValue",1,5);

System.out.println(count);

5、range(K key, long start, long end) 获取变量指定区间的元素

Set valueSet = redisTemplate.opsForZSet().range("value",0,-1);

System.out.println(valueSet);

6、rangeByLex(K key, RedisZSetCommands.Range range) 用于获取满足非score的排序取值。

这个排序只有在有相同分数的情况下才能使用，如果有不同的分数则返回值不确定。

RedisZSetCommands.Range range = new RedisZSetCommands.Range();

//range.gt("A");

range.lt("B");

Set valueSet = redisTemplate.opsForZSet().rangeByLex("value", range);

System.out.println(valueSet);

7、rangeByScore(K key, double min, double max)

//根据设置的score获取区间值。

Set valueSet = redisTemplate.opsForZSet().rangeByScore("zSetValue",1,2);

System.out.println(valueSet);

8、incrementScore(K key, V value, double delta) 修改元素的分值。

double incrementScore = redisTemplate.opsForZSet().incrementScore("zSetValue","A",5);

//获取元素的分值

score = redisTemplate.opsForZSet().score("zSetValue","A");

System.out.println("修改后A的分值:" + score);

9、rangeByScoreWithScores(K key, double min, double max)

正序获取RedisZSetCommands.Tuples的区间值通过分值。

Set<ZSetOperations.TypedTuple<Object>> typedTupleSet = redisTemplate.opsForZSet().rangeByScoreWithScores("typedTupleSet",5,8);

iterator = typedTupleSet.iterator();

while (iterator.hasNext()){

ZSetOperations.TypedTuple<Object> typedTuple = iterator.next();

Object value = typedTuple.getValue();

double score = typedTuple.getScore();

System.out.println(value + "----" + score );

}

10.reverseRangeByScoreWithScores(K key, double min, double max)

倒序排序获取RedisZSetCommands.Tuples的分值区间值。

Set<ZSetOperations.TypedTuple<Object>> typedTupleSet = redisTemplate.opsForZSet().reverseRangeByScoreWithScores("zSetValue",1,5);

iterator = typedTupleSet.iterator();

while (iterator.hasNext()){

ZSetOperations.TypedTuple<Object> typedTuple = iterator.next();

Object value = typedTuple.getValue();

double score = typedTuple.getScore();

System.out.println(value + "----" + score);

}