Redies

Redies启动命令

```
E:\ItApp\Redis-x64-3.2.100>redis-server.exe redis.windows.conf
```

检测是否连接成功

```
keys *
```

Redis的基本数据类型

```
Redis存储的是key-value结构的数据，其中key是字符串类型，value有5种常用的数据类型
1:字符串 string Redis最简单的数据类型
2:哈希 hash 也叫散列，类似于java中的HashMao结构
3:列表 list 按照插入顺序排表，可以有重复元素，类似于java中的LinkedList
4:集合 set 无序集合，没有重复元素，类似于java中的HashSet
5:有序集合 sorted set/zset 集合中每个元素关联一个分数(score),根据分数升序排序，没有重复元素
```

Redis的常用命令

```
字符串操作命令
1: set key value 设置指定key的值 eg：set name jack 为name设置值为jack
2: get key 获取指定key的值 eg: get name 得到name的值
3: setex key seconds value 设置指定key的值,并将key过期时间设为seconds秒 eg:setex code
30 12345
4: setnx key value 只有在key不存在时设置key eg:setnx key1 itcast 再使用setnx key1
itheima不能够使用
```

```
哈希操作命令
1:HSET key field value 将哈希表key中的字段field的值设为value eg:HSET 100 name
xiaoming
2:HGET key field 获取存储在哈希表中指定字段的值 eg:hget 100 name
3:HDEL key field 删除存储在哈希表中的指定字段 eg:hdEL 100 name
4:HKEYS key 获取哈希表中所有字段 eg：HKEYS 100
5:HVALS key 获取哈希表中所有的值 eg： HVALS 100
```

```
列表操作命令
1:LPUSH key value[values] 将一个或多个值插入到列表的头部 eg: lpush key a b c
2:LRANGE key strat stop 获取列表指定范围内的元素eg: lrange key 1 -1 查询全部元素
3:RPOP key 移除并获取列表最后一个元素从右侧 eg: RPOP key
4:LLEN key 获取列表的元素 eg: LLEN RPOP
```

```
集合操作命令
1:SADD key member1[memebr2] 向集合添加一个或多个成员eg:sadd set1 a b c d
2:SMEMBERS key 返回集中中的所有成员
3:SCARD key 获取集合的成员数
4:SINTER key1 [key2] 返回给定所有集合的交集
5:SUNION key1 [key2] 返回所有给定集合的并集
6:SREM key member1 [member2] 删除集合中一个或多个成员
```

```
有序集合操作命令
1:ZADD key score1 member1[score1 member1]向有序集合添加一个或多个成员
2:ZRANGE key start stop[WITHSCORES]通过索引区间返回有序集合中指定区间内成员
3:ZINCRBY key increment member 有序集合中对指定成员的分数加上增量 increment
4:ZREM key member [member] 移除有序集合中的一个或多个成员
```

```
通用命令
1:KEYS pettren 查找所有符合给定模式(pattern)的Key
2:EXISTS key 检查给定key是否存在
3:TYPE key 返回key所储存的值的类型
4:DEL key 用于在key存在时删除key
```

Java中操作radis

```
Redis的Java客户端
1:Jedis
2:Lettuce
3:Spring Data Redis 是spring的一部分，对Redis底层开发包进行了高度封装.
```

Spring Data Redis使用方式

```
1:导入maven坐标
<dependency>
  <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

```
2：配置Redis数据源
  redis:
    host: localhost #连接地址
    port: 6379 #端口号
    database: 0 #数据库默认为0
```

```
3：编写配置类创建RedisTemplate对象
package com.sky.config;
import lombok.extern.slf4j.Slf4j;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.data.redis.serializer.StringRedisSerializer;
@Configuration
@Slf4j
public class RedisConfiguration {
```

```
    @Bean
    public RedisTemplate redisTemplate(RedisConnectionFactory
redisConnectionFactory){
        log.info("开始创建redis模板对象");
        RedisTemplate redisTemplate=new RedisTemplate<>();
        //设置redis的连接工厂对象
        redisTemplate.setConnectionFactory(redisConnectionFactory);
        //设置redis key的序列化器
        redisTemplate.setKeySerializer(new StringRedisSerializer());
        return  redisTemplate;
    }
}
```

4：通过RedisTemplate对象操作Redis

可以存入任意对象，最终都转为string存储

字符串类型：

```
   /*
    * 操作字符串类型
    * */
    @Test
    public  void testString(){
        //set get setex setnx
        redisTemplate.opsForValue().set("city","北京");
        String city = (String) redisTemplate.opsForValue().get("city");
        System.out.println(city);
        //setnx
        redisTemplate.opsForValue().set("code","1234",3, TimeUnit.MINUTES);
        redisTemplate.opsForValue().setIfAbsent("lock","1");
        redisTemplate.opsForValue().setIfAbsent("lock","2");
    }
```

哈希类型：

```
   /*
    * 哈希类型
    * */
    @Test
    public void testHash(){
        //hset hget hdel hkeys hyals
        HashOperations hashOperations = redisTemplate.opsForHash();
        hashOperations.put("100","name","tom");
        hashOperations.put("100","age","20");
        String name = (String) hashOperations.get("100", "name");
        System.out.println(name);
        Set keys = hashOperations.keys("100");
        System.out.println(keys);
        List values = hashOperations.values("100");
        System.out.println(values);
        hashOperations.delete("100","age");
    }
```

列表数据：

```
    /*
    * 操作列表类型的数据
    * */
    @Test
    public void textList(){
        ListOperations listOperations = redisTemplate.opsForList();
        listOperations.leftPushAll("mylist","a","b","c","d");//插入多条数据
        listOperations.leftPush("mylist","d");//插入一条数据
```

```
        List mylist = listOperations.range("mylist", 0, -1);//查询范围内的值
        System.out.println(mylist);
        listOperations.rightPop("mylist");//从右侧获得数据
        Long size = listOperations.size("mylist");//获取列表的元素
        System.out.println(size);
    }
```

集合类型：

```
  /*
    * 操作集合
    * */
    @Test
    public void testSet(){
        SetOperations setOperations = redisTemplate.opsForSet();
        setOperations.add("set1","a","b","c","d");
        setOperations.add("set2","a","b","c","d");
        Set members = setOperations.members("set1");
        System.out.println(members);
        Long size = setOperations.size("set1");
        System.out.println(size);
        Set intersect = setOperations.intersect("set1","set2");
        System.out.println(intersect);
        Set union = setOperations.union("set1", "Set2");
        System.out.println(union);
        setOperations.remove("set1","a","b");
    }
```

有序集合：

```
   /*
    * 有序集合
    * */
    @Test
    public void testZset(){
        ZSetOperations zSetOperations = redisTemplate.opsForZSet();
        zSetOperations.add("zset1","a",10);
        zSetOperations.add("zset1","b",12);
        zSetOperations.add("zset1","c",9);
        Set zset1 = zSetOperations.range("zset1", 0, -1);
        System.out.println(zset1);
        zSetOperations.incrementScore("zset1","c",10);
    }
```

通用指令操作：

```
  /*
    * 通用命令操作
    * */
    @Test
    public void testCommon(){
        Set keys = redisTemplate.keys("*");//查询所有的keys
        System.out.println(keys);
        Boolean name = redisTemplate.hasKey("name");
        Boolean set1 = redisTemplate.hasKey("set1");
        //遍历key
        for (Object key:keys){
            DataType type=redisTemplate.type(key);
            System.out.println(type.name());
        }
       redisTemplate.delete("mylist");
    }
```