Redis 常用操作命令，非常详细！

下面总结并演示了 Redis 的 常用管理命令、key 操作、字符串、集合、列表、散列类型的操作命令。

**常用管理命令**

**1、启动Redis**

> redis-server [--port 6379]

如果命令参数过多，建议通过配置文件来启动Redis。

> redis-server [xx/xx/redis.conf]

6379是Redis默认端口号。

**2、连接Redis**

> ./redis-cli [-h 127.0.0.1 -p 6379]

**3、停止Redis**

> redis-cli shutdown  
  
> kill redis-pid

以上两条停止Redis命令效果一样。

**4、发送命令**

给Redis发送命令有两种方式：

1、redis-cli带参数运行，如：

> redis-cli shutdown  
not connected>

这样默认是发送到本地的6379端口。

2、redis-cli不带参数运行，如：

> ./redis-cli  
  
127.0.0.1:6379> shutdown  
not connected>

**5、测试连通性**

127.0.0.1:6379> ping  
PONG

**key操作命令**

**获取所有键**

语法：keys pattern

127.0.0.1:6379> keys \*  
1) "javastack"

* \*表示通配符，表示任意字符，会遍历所有键显示所有的键列表，时间复杂度O(n)，在生产环境不建议使用。

**获取键总数**

语法：dbsize

127.0.0.1:6379> dbsize  
(integer) 6

获取键总数时不会遍历所有的键，直接获取内部变量，时间复杂度O(1)。

**查询键是否存在**

语法：exists key [key …]

127.0.0.1:6379> exists javastack java  
(integer) 2

查询查询多个，返回存在的个数。

**删除键**

语法：del key [key …]

127.0.0.1:6379> del java javastack  
(integer) 1

可以删除多个，返回删除成功的个数。

**查询键类型**

语法： type key

127.0.0.1:6379> type javastack  
string

**移动键**

语法：move key db

如把javastack移到2号数据库。

127.0.0.1:6379> move javastack 2  
(integer) 1  
127.0.0.1:6379> select 2  
OK  
127.0.0.1:6379[2]> keys \*  
1) "javastack"

**查询key的生命周期（秒）**

秒语法：ttl key

毫秒语法：pttl key

127.0.0.1:6379[2]> ttl javastack  
(integer) -1

-1：永远不过期。

**设置过期时间**

秒语法：expire key seconds

毫秒语法：pexpire key milliseconds

127.0.0.1:6379[2]> expire javastack 60  
(integer) 1  
127.0.0.1:6379[2]> ttl javastack  
(integer) 55

**设置永不过期**

语法：persist key

127.0.0.1:6379[2]> persist javastack  
(integer) 1

**更改键名称**

语法：rename key newkey

127.0.0.1:6379[2]> rename javastack javastack123  
OK

**字符串操作命令**

字符串是Redis中最基本的数据类型，单个数据能存储的最大空间是512M。

**存放键值**

语法：set key value [EX seconds] [PX milliseconds] [NX|XX]

nx：如果key不存在则建立，xx：如果key存在则修改其值，也可以直接使用setnx/setex命令。

127.0.0.1:6379> set javastack 666  
OK

**获取键值**

语法：get key

127.0.0.1:6379[2]> get javastack  
"666"

**值递增/递减**

如果字符串中的值是数字类型的，可以使用incr命令每次递增，不是数字类型则报错。

语法：incr key

127.0.0.1:6379[2]> incr javastack  
(integer) 667

一次想递增N用incrby命令，如果是浮点型数据可以用incrbyfloat命令递增。

同样，递减使用decr、decrby命令。

**批量存放键值**

语法：mset key value [key value …]

127.0.0.1:6379[2]> mset java1 1 java2 2 java3 3  
OK

**获取获取键值**

语法：mget key [key …]

127.0.0.1:6379[2]> mget java1 java2  
1) "1"  
2) "2"

Redis接收的是UTF-8的编码，如果是中文一个汉字将占3位返回。

**获取值长度**

语法：strlen key

127.0.0.1:6379[2]> strlen javastack  
(integer) 3

**追加内容**

语法：append key value

127.0.0.1:6379[2]> append javastack hi  
(integer) 5

向键值尾部添加，如上命令执行后由666变成666hi

**获取部分字符**

语法：getrange key start end

> 127.0.0.1:6379[2]> getrange javastack 0 4  
"javas"

**集合操作命令**

集合类型和列表类型相似，只不过是集合是无序且不可重复的。

**集合**

**存储值**

语法：sadd key member [member …]

// 这里有8个值（2个java），只存了7个  
127.0.0.1:6379> sadd langs java php c++ go ruby python kotlin java  
(integer) 7

**获取元素**

获取所有元素语法：smembers key

127.0.0.1:6379> smembers langs  
1) "php"  
2) "kotlin"  
3) "c++"  
4) "go"  
5) "ruby"  
6) "python"  
7) "java"

随机获取语法：srandmember langs count

127.0.0.1:6379> srandmember langs 3  
1) "c++"  
2) "java"  
3) "php"

**判断集合是否存在元素**

语法：sismember key member

127.0.0.1:6379> sismember langs go  
(integer) 1

**获取集合元素个数**

语法：scard key

127.0.0.1:6379> scard langs  
(integer) 7

**删除集合元素**

语法：srem key member [member …]

127.0.0.1:6379> srem langs ruby kotlin  
(integer) 2

**弹出元素**

语法：spop key [count]

127.0.0.1:6379> spop langs 2  
1) "go"  
2) "java"

**有序集合**

和列表的区别：

1、列表使用链表实现，两头快，中间慢。有序集合是散列表和跳跃表实现的，即使读取中间的元素也比较快。

2、列表不能调整元素位置，有序集合能。

3、有序集合比列表更占内存。

**存储值**

语法：zadd key [NX|XX] [CH] [INCR] score member [score member …]

127.0.0.1:6379> zadd footCounts 16011 tid 20082 huny 2893 nosy  
(integer) 3

**获取元素分数**

语法：zscore key member

127.0.0.1:6379> zscore footCounts tid  
"16011"

获取排名范围排名语法：zrange key start stop [WITHSCORES]

// 获取所有，没有分数  
127.0.0.1:6379> zrange footCounts 0 -1  
1) "nosy"  
2) "tid"  
3) "huny"  
  
// 获取所有及分数  
127.0.0.1:6379> zrange footCounts 0 -1 Withscores  
1) "nosy"  
2) "2893"  
3) "tid"  
4) "16011"  
5) "huny"  
6) "20082"

获取指定分数范围排名语法：zrangebyscore key min max [WITHSCORES] [LIMIT offset count]

127.0.0.1:6379> zrangebyscore footCounts 3000 30000 withscores limit 0 1  
1) "tid"  
2) "16011"

**增加指定元素分数**

语法：zincrby key increment member

127.0.0.1:6379> zincrby footCounts 2000 tid  
"18011"

**获取集合元素个数**

语法：zcard key

127.0.0.1:6379> zcard footCounts  
(integer) 3

**获取指定范围分数个数**

语法：zcount key min max

127.0.0.1:6379> zcount footCounts 2000 20000  
(integer) 2

**删除指定元素**

语法：zrem key member [member …]

127.0.0.1:6379> zrem footCounts huny  
(integer) 1

**获取元素排名**

语法：zrank key member

127.0.0.1:6379> zrank footCounts tid  
(integer) 1

**列表操作命令**

列表类型是一个有序的字段串列表，内部是使用双向链表实现，所有可以向两端操作元素，获取两端的数据速度快，通过索引到具体的行数比较慢。

列表类型的元素是有序且可以重复的。

**存储值**

左端存值语法：lpush key value [value …]

127.0.0.1:6379> lpush list lily sandy  
(integer) 2

右端存值语法：rpush key value [value …]

127.0.0.1:6379> rpush list tom kitty  
(integer) 4

索引存值语法：lset key index value

127.0.0.1:6379> lset list 3 uto  
OK

**弹出元素**

左端弹出语法：lpop key

127.0.0.1:6379> lpop list  
"sandy"

右端弹出语法：rpop key

127.0.0.1:6379> rpop list  
"kitty"

**获取元素个数**

语法：llen key

127.0.0.1:6379> llen list  
(integer) 2

**获取列表元素**

两边获取语法：lrange key start stop

127.0.0.1:6379> lpush users tom kitty land pony jack maddy  
(integer) 6  
  
127.0.0.1:6379> lrange users 0 3  
1) "maddy"  
2) "jack"  
3) "pony"  
4) "land"  
  
// 获取所有  
127.0.0.1:6379> lrange users 0 -1  
1) "maddy"  
2) "jack"  
3) "pony"  
4) "land"  
5) "kitty"  
6) "tom"  
  
// 从右端索引  
127.0.0.1:6379> lrange users -3 -1  
1) "land"  
2) "kitty"  
3) "tom"

索引获取语法：lindex key index

127.0.0.1:6379> lindex list 2  
"ketty"  
  
// 从右端获取  
127.0.0.1:6379> lindex list -5  
"sady"

**删除元素**

根据值删除语法：lrem key count value

127.0.0.1:6379> lpush userids 111 222 111 222 222 333 222 222  
(integer) 8  
  
// count=0 删除所有  
127.0.0.1:6379> lrem userids 0 111  
(integer) 2  
  
// count > 0 从左端删除前count个  
127.0.0.1:6379> lrem userids 3 222  
(integer) 3  
  
// count < 0 从右端删除前count个  
127.0.0.1:6379> lrem userids -3 222  
(integer) 2

范围删除语法：ltrim key start stop

// 只保留2-4之间的元素  
127.0.0.1:6379> ltrim list 2 4  
OK

**散列操作命令**

redis字符串类型键和值是字典结构形式，这里的散列类型其值也可以是字典结构。

**存放键值**

单个语法：hset key field value

127.0.0.1:6379> hset user name javastack  
(integer) 1

多个语法：hmset key field value [field value …]

127.0.0.1:6379> hmset user name javastack age 20 address china  
OK

不存在时语法：hsetnx key field value

127.0.0.1:6379> hsetnx user tall 180  
(integer) 0

**获取字段值**

单个语法：hget key field

127.0.0.1:6379> hget user age  
"20"

多个语法：hmget key field [field …]

127.0.0.1:6379> hmget user name age address  
1) "javastack"  
2) "20"  
3) "china"

获取所有键与值语法：hgetall key

127.0.0.1:6379> hgetall user  
1) "name"  
2) "javastack"  
3) "age"  
4) "20"  
5) "address"  
6) "china"

获取所有字段语法：hkeys key

127.0.0.1:6379> hkeys user  
1) "name"  
2) "address"  
3) "tall"  
4) "age"

获取所有值语法：hvals key

127.0.0.1:6379> hvals user  
1) "javastack"  
2) "china"  
3) "170"  
4) "20"

**判断字段是否存在**

语法：hexists key field

127.0.0.1:6379> hexists user address  
(integer) 1

**获取字段数量**

语法：hlen key

127.0.0.1:6379> hlen user  
(integer) 4

**递增/减**

语法：hincrby key field increment

127.0.0.1:6379> hincrby user tall -10  
(integer) 170

**删除字段**

语法：hdel key field [field …]

127.0.0.1:6379> hdel user age  
(integer) 1