Redis简介

Redis是一个开源的，高级的键值存储和一个适用的解决方案，用于构建高性能，可扩展的Web应用程序。Redis也被作家戏称为数据结构服务器；

Redis优点：

1. **异常的快**：Redis非常快，每秒大约可执行110000次设置操作，81000次读取操作；
2. **支持丰富的数据类型**：Redis支持开发人员常用的大多数数据结构，String，集合，列表等；
3. **操作具有原子性**：所有Redis操作都是原子操作，确保客户端并发访问，Redis接收更新的值；
4. **多实用工具**：Redis是一个多实用工具；

Redis五种基本数据结构

string，hash，list，set，zset；

每种数据结构多种底层编码实现，每种底层编码可能同时支持多种数据结构；

字符串string

动态字符串，使用者可以修改，底层类似于ArrayList，有一个字符数组；Redis 底层对于字符串的定义 SDS，即 Simple Dynamic String 结构：

字符串比较短的时候，len和alloc可以使用byte和short来表示，Redis为了对内存有极致的优化，不同长度字符串使用不同的结构体；

对字符串的基本操作

设置和获取键值对：

```
> SET key value
OK
> GET key
"value"
```

值可以是任何种类的字符串（包括二进制数据），例如你可以在一个键下保存一张 .jpeg 图片，只需要注意不要超过 512 MB 的最大限度就好了。

当key存在的时候SET操作覆盖上次的值；

可以用`EXISTS和DEL`关键字来删除键值对；

批量设置键值对：

```
MSET key1 value1 key2 value2
MGET key1 key2
```

过期和SET命令扩展：

```
EXPIRE name 5

等价于SET + EXPIRE 的 SETEX命令
SETEX key 5 value1
```

计数 如果value是一个整数的话，还可以对他使用INCR命令原子性的自增操作，这意味着及时多个客户端对同一个 key 进行操作，也决不会导致竞争的情况：

```
> SET counter 100
> INCR counter
(integer) 101
> INCRBY counter 50
(integer) 151
```

返回原值的GETSET命令 它的功能跟它名字一样：为 key 设置一个值并返回原值：

列表list

Redis 的列表相当于 Java 语言中的 LinkedList，注意它是链表而不是数组。这意味着 list 的插入和删除操作非常快，时间复杂度为 O(1)，但是索引定位很慢，时间复杂度为 O(n)。 双向链表；

链表的基本操作 LPUSH 和 ROUSH 分别可以向list的左边头部和右边尾部添加一个新元素； LRANGE 命令可以从list中取出一定的范围的元素； LINDEX 命令可以从list中取出指定的元素，相当于Java链表操作中get(int index)操作；

```
> rpush mylist A
(integer) 1
> rpush mylist B
(integer) 2
> lpush mylist first
(integer) 3
> lrange mylist 0 -1    # -1 表示倒数第一个元素，这里表示从第一个元素到最后一个元素，即所有
1) "first"
2) "A"
3) "B"
```

list实现队列 先进先出：

```
> RPUSH books python java golang
(integer) 3
> LPOP books
"python"
> LPOP books
"java"
> LPOP books
"golang"
> LPOP books
(nil)
```

list实现栈 先进后出：

```
> RPUSH books python java golang
> RPOP books
"golang"
> RPOP books
"java"
> RPOP books
"python"
> RPOP books
(nil)
```

两种操作添加操作都是RPUSH的操作，出栈/队操作不一样，队列出LPOP，栈出RPOP；

字典hash

Redis 中的字典相当于 Java 中的 HashMap，内部实现也差不多类似，都是通过 "数组 + 链表" 的链地址法来解决部分 哈希冲突，同时这样的结构也吸收了两种不同数据结构的优点。

实际上字典结构中包含两个hashtable，通常只有一个hashtable有值，但是字典扩容缩容需要分配新的hashtable然后渐进式搬迁；

渐进式rehash 渐进式Rehash，渐进式会在rehash的同时保留两个hash结构，查询时同时查询两个hash结构，然后再侯勋的定时任务或者hash指令中，逐渐的把旧字典内容迁移到新字典中。

扩缩容条件 正常情况下，hash表元素的个数**等于第一组数组的长度时**，开始扩容，扩容数组是元素组大小的2倍，不过如果Redis正在做bgsave，为了减少内存也得过多分离，Redis尽量不去扩容。如果hash非常满了，达到了第一维数组的5倍了就会强制扩容；

当 hash 表因为元素逐渐被删除变得越来越稀疏时，Redis 会对 hash 表进行缩容来减少 hash 表的第一维数组空间占用。所用的条件是 元素个数低于数组长度的 10%，缩容不会考虑 Redis 是否在做 bgsave。

```
> HSET books java "think in java"    # 命令行的字符串如果包含空格则需要使用引号包裹
(integer) 1
> HSET books python "python cookbook"
```

```
(integer) 1
> HGETALL books    # key 和 value 间隔出现
1) "java"
2) "think in java"
3) "python"
4) "python cookbook"
> HGET books java
"think in java"
> HSET books java "head first java"
(integer) 0        # 因为是更新操作，所以返回 0
> HMSET books java "effetive  java" python "learning python"    # 批量操作
OK
```

集合Set

Redis 的集合相当于 Java 语言中的 HashSet，它内部的键值对是无序、唯一的。它的内部实现相当于一个特殊的字典，字典中所有的 value 都是一个值 NULL。

```
> SADD books java
(integer) 1
> SADD books java    # 重复
(integer) 0
> SADD books python golang
(integer) 2
> SMEMBERS books    # 注意顺序，set 是无序的
1) "java"
2) "python"
3) "golang"
> SISMEMBER books java    # 查询某个 value 是否存在，相当于 contains
(integer) 1
> SCARD books    # 获取长度
(integer) 3
> SPOP books     # 弹出一个
"java"
```

有序列表zset

具有特色的一个数据结构，类似于java的SortedSet和HashMap的结合体，一方面是一个set保持唯一性，一方面保证了内部value的唯一性，可以为每一个value赋予一个score值，用来代表排序的权重；

```
> ZADD books 9.0 "think in java"
> ZADD books 8.9 "java concurrency"
> ZADD books 8.6 "java cookbook"

> ZRANGE books 0 -1     # 按 score 排序列出，参数区间为排名范围
1) "java cookbook"
2) "java concurrency"
```

```
3) "think in java"

> ZREVRANGE books 0 -1  # 按 score 逆序列出，参数区间为排名范围
1) "think in java"
2) "java concurrency"
3) "java cookbook"

> ZCARD books           # 相当于 count()
(integer) 3

> ZSCORE books "java concurrency"   # 获取指定 value 的 score
"8.9000000000000004"                # 内部 score 使用 double 类型进行存储，所以存在小数点精度问题

> ZRANK books "java concurrency"    # 排名
(integer) 1

> ZRANGEBYSCORE books 0 8.91        # 根据分值区间遍历 zset
1) "java cookbook"
2) "java concurrency"

> ZRANGEBYSCORE books -inf 8.91 withscores  # 根据分值区间 (-∞, 8.91] 遍历 zset，同时返回分值。inf 代表 infinite，无穷大的意思。
1) "java cookbook"
2) "8.5999999999999996"
3) "java concurrency"
4) "8.9000000000000004"

> ZREM books "java concurrency"             # 删除 value
(integer) 1
> ZRANGE books 0 -1
1) "java cookbook"
2) "think in java"
```