**Redis,Memcached学习笔记**

**V1.0.0**

**Author: clearuo**

Date: 2016-04-06

# Redis文档

## 一 介绍

### 1.1 可视化管理工具

Redis Desktop Manager

### 1.2 简单介绍

Redis是一个开源的，先进的 key-value 存储可用于构建高性能，可扩展的 Web 应用程序的解决方案。Redis官方网网站是：<http://www.redis.io/>

### 1.3 Redis特点

Redis是完全开源免费的, 遵守BSD协议, 是一个高性能的key-value数据库.

Redis与其它key-value缓存数据库产品有以下三个特点：

* Redis支持数据的持久化, 可以将内存中的数据保存在磁盘中, 重启的时候可以再次加载进行使用；
* Redis不仅仅支持简单的key-value类型数据库, 同时还提供list, set, zset, hash等数据结构的存储；
* Redis支持数据库的备份, 即master-slave模式的数据备份；

### 1.4 Redis优点

* **性能极高**: Redis读的速度是110000次/s, 写的速度是81000次/s；
* **丰富的数据类型**: Redis支持二进制案例的Strings, Lists, Hashes, Sets及Ordered Sets数据类型操作;
* **原子性**: Redis的所有操作都是原子性的, 同时Redis还支持对几个操作全并后的院子性执行.
* **丰富的特性**: Redis还支持publish/subscribe, 通知, key过期等等特性;

### 1.5 Redis与其它key-value存储有什么不同?

* Redis有着更为复杂的数据结构并且提供对他们的原子性操作,这是一个不同于其它数据库的进化路径.Redis的数据类型都是基于基本数据结构的同时对程序员透明,无需进行额外的抽象.
* Redis运行在内存中但是可以持久化到磁盘,所以在对不同数据集进行高速读写时需要权衡内存,因为数据量不能大于硬件内存.在内存数据库方面的另一个优点是,相比在磁盘上相同的复杂的数据结构,在内存中操作起来非常简单,这样Redis可以做很多内部复杂性很强的事情.同时,在磁盘格式方面他们是紧凑的以追加的方式产生的,因为他们并不需要进行随机访问.

### 1.6 Redis环境

要在 Ubuntu 上安装 Redis，打开终端，然后输入以下命令：

|  |
| --- |
| $sudo apt-get update $sudo apt-get install redis-server |

这将在您的计算机上安装Redis

启动 Redis

|  |
| --- |
| $redis-server |

查看 redis 是否还在运行

这将打开一个 Redis 提示符，如下图所示：



在上面的提示信息中：127.0.0.1 是本机的IP地址，6379是 Redis 服务器运行的端口。现在输入 PING 命令，如下图所示：



这说明现在你已经成功地在计算机上安装了 Redis。

* 远程服务器服务器上运行命令

语法:

|  |
| --- |
| $ redis-cli -h host -p port -a password |

例如:

下面的示例演示了如何连接到Redis主机：127.0.0.1，端口：6379 上的远程服务器，并加上验证密码为：mypass

|  |
| --- |
| $redis-cli -h 127.0.0.1 -p 6379 -a "mypass" redis 127.0.0.1:6379> redis 127.0.0.1:6379> PING PONG |

### 1.7 Redis配置

在Redis有配置文件(redis.conf)可在Redis的根目录下找到。可以通过Redis的CONFIG命令设置所有Redis的配置。

* 语法

Redis的CONFIG命令的基本语法如下所示：

|  |
| --- |
| redis 127.0.0.1:6379> CONFIG GET CONFIG\_SETTING\_NAME |

* 例如

|  |
| --- |
| redis 127.0.0.1:6379> CONFIG GET loglevel 1) "loglevel" 2) "notice" |

* 编辑配置

要更新配置，可以直接编辑redis.conf文件或更新配置，通过CONFIG set命令

* 语法

CONFIG SET命令的基本语法如下所示：

|  |
| --- |
| redis 127.0.0.1:6379> CONFIG SET CONFIG\_SETTING\_NAME NEW\_CONFIG\_VALUE |

* 例子

|  |
| --- |
| redis 127.0.0.1:6379> CONFIG SET loglevel "notice" OK redis 127.0.0.1:6379> CONFIG GET loglevel 1) "loglevel" 2) "notice" |

## 二 Redis数据类型

Redis支持5种数据类型,它们描述如下:

### 2.1 String(字符串)

string是redis最基本的类型,你可以理解成与Memcached一模一样的类型,一个key对应一个value.

String类型是二进制安全的,意思是redis的string可以包含任何数据.比如jpg图片或者序列号的对象.

String类型是redis最基本的数据类型,一个键最大能存储512MB.

例子:

|  |
| --- |
| redis 127.0.0.1:6379> SET name "yiibai" OK redis 127.0.0.1:6379> GET name "yiibai" |

在上面的例子使用Redis命令set和get，Redis的名称是yiibai的键存储在Redis的字符串值。

**注**：字符串值可以存储最大512兆字节的长度。

### 2.2 Hash(哈希值)

Redishash是键值对的集合。

Redis hash 是一个string类型的field和value的映射表,hash特别适合用于存储对象.

例子:

|  |
| --- |
| redis 127.0.0.1:6379> HMSET user:1 username yiibai password yiibai points 200 OK redis 127.0.0.1:6379> HGETALL user:1 1) "username" 2) "yiibai" 3) "password" 4) "yiibai" 5) "points" 6) "200" |

在上面的例子中的哈希数据类型，用于存储包含用户的基本信息用户的对象。这里HMSET，HEXTALL对于Redis 命令 user:1 是键。

每个哈希可存储多达2^32 - 1个 键值对(超过四十亿)。

### 2.3 List(列表)

Redis的列表是简单的字符串列表，按照插入顺序排序。可以添加元素到Redis列表的头部或尾部。

例子:

|  |
| --- |
| redis 127.0.0.1:6379> lpush tutoriallist redis (integer) 1 redis 127.0.0.1:6379> lpush tutoriallist mongodb (integer) 2 redis 127.0.0.1:6379> lpush tutoriallist rabitmq (integer) 3 redis 127.0.0.1:6379> lrange tutoriallist 0 10 1) "rabitmq" 2) "mongodb" 3) "redis" |

列表中最大长度为2^32 - 1(每个列表成员超过四十亿)。

### 2.4 Set(集合)

Redis集合是string类型的无序集合。

结合是通过哈希表实现的,所以添加,删除,查找的复杂度都是O(1).

例子

|  |
| --- |
| redis 127.0.0.1:6379> sadd tutoriallist redis (integer) 1 redis 127.0.0.1:6379> sadd tutoriallist mongodb (integer) 1 redis 127.0.0.1:6379> sadd tutoriallist rabitmq (integer) 1 redis 127.0.0.1:6379> sadd tutoriallist rabitmq (integer) 0 redis 127.0.0.1:6379> smembers tutoriallist 1) "rabitmq" 2) "mongodb" 3) "redis" |

注意：在上面的例子中rabitmq设置属性加两次，但由于唯一性只加一次。

成员中集最大数量为2^32 - 1(4294967295，集合成员超过4十亿)。

### 2.5 ZSET(sorted set:有序集合)

* Redis zset和set一样也是string类型元素的集合,且不允许重复的成员.
* 不同的是每个元素都会关联一个double类型的分数.redis正是通过分数来为集合中的成员进行从小到大的排序.
* Zset的成员是唯一的,但分数(score)却可以重复
* 实例

|  |
| --- |
| redis 127.0.0.1:6379> zadd tutoriallist 0 redis (integer) 1 redis 127.0.0.1:6379> zadd tutoriallist 0 mongodb (integer) 1 redis 127.0.0.1:6379> zadd tutoriallist 0 rabitmq (integer) 1 redis 127.0.0.1:6379> zadd tutoriallist 0 rabitmq (integer) 0 redis 127.0.0.1:6379> ZRANGEBYSCORE tutoriallist 0 1000  1) "redis" 2) "mongodb" 3) "rabitmq" |

## 三 Redis键

Redis的keys命令用于管理键,使用Redis的keys命令语法如下所示:

语法

|  |
| --- |
| redis 127.0.0.1:6379> COMMAND KEY\_NAME |

例子

|  |
| --- |
| redis 127.0.0.1:6379> SET yiibai redis OK redis 127.0.0.1:6379> DEL yiibai (integer) 1 |

在上面的例子中DEL是命令,而yiibai是键,如果键被删除那么输出该命令将是(integer)1,否则它是(integer)0

### 3.1 DEL key

* 功能:
* Redis的DEL命令用于删除redis中现有的键,如果存在.
* 返回值:
* 被删除的键的数目
* 语法:

|  |
| --- |
| redis 127.0.0.1:6379> DEL KEY\_NAME |

* 例子
* 首先,在Redis的一个键,并设置一定的值

|  |
| --- |
| redis 127.0.0.1:6379> SET yiibai redis OK |

* 现在删除以前生成的键

|  |
| --- |
| redis 127.0.0.1:6379> DEL yiibai (integer) 1 |

### 3.2 DUMP key

* 功能:
* Redis的DUMP命令用于获取存储在redis的指定键的数据的序列化版本.
* 返回值:
* 序列化的值(字符串)
* 语法:

|  |
| --- |
| redis 127.0.0.1:6379> DUMP KEY\_NAME |

* 例子:
* 首先,在Redis创建一个键,并设置一定的值.

|  |
| --- |
| redis 127.0.0.1:6379> SET yiibai redis OK |

* 现在获取之前创建的键的值的序列化

|  |
| --- |
| redis 127.0.0.1:6379> DUMP yiibai "\x00\x05redis\x06\x00S\xbd\xc1q\x17z\x81\xb2" |

### 3.3 EXISTS key

* 功能:
* Redis EXISTS命令被用来检查键是否存在与redis.
* 返回值:
* 1-如果键存在
* 0-如果键不存在
* 语法:

|  |
| --- |
| redis 127.0.0.1:6379> EXISTS KEY\_NAME |

* 例子:

|  |
| --- |
| redis 127.0.0.1:6379> EXISTS yiibai-new-key (integer) 0 |

* 现在,创建一个名为yiibai新键,并检查是否存在

|  |
| --- |
| 127.0.0.1:6379> SET yiibai-new-key 'hahafds' OK 127.0.0.1:6379> EXISTS yiibai-new-key (integer) 1 127.0.0.1:6379> |

### 3.4 EXPIRE key seconds

* 功能:
* Redis Expire命令用于设定键有效期,到期时间后键将失效.
* 返回值:
* 1-如果设置成功.
* 0-如果键不存在,或者设置失败.
* 语法:

|  |
| --- |
| redis 127.0.0.1:6379> Expire KEY\_NAME TIME\_IN\_SECONDS |

* 例子:
* 首先,在Redis创建一个键,并设置一定的值

|  |
| --- |
| redis 127.0.0.1:6379> SET yiibai redis OK |

* 现在设置以前创建的键超时时间

|  |
| --- |
| redis 127.0.0.1:6379> EXPIRE yiibai 60 (integer) 1 |

* 在上面的例子中键yiibai被设定60秒的时间,60秒后,键会自动失效

### 3.5 EXPIREAT key timestamp

* 功能:
* Redis Exipreat命令是用来以Unix时间戳格式设置键的到期时间, 到期后键将失效.
* 返回值:
* 1-如果设置成功.
* 0-如果键不存在,或者设置失败.
* 语法:

|  |
| --- |
| redis 127.0.0.1:6379> Expireat KEY\_NAME TIME\_IN\_UNIX\_TIMESTAMP |

* 例子:
* 首先,在Redis创建一个键,并设置一定的值.

|  |
| --- |
| redis 127.0.0.1:6379> SET yiibai redis OK |

* 现在设置以前创建的键超时

|  |
| --- |
| redis 127.0.0.1:6379> EXPIREAT yiibai 1482817430 (integer) 1 EXISTS yiibai (integer) 0 |

### 3.6 PEXPIRE key milliseconds

* 功能:
* 和EXPIRE键类似,设置键以毫秒为单位到期
* 返回值:
* 语法:
* 例子:

### 3.7 PEXPIREAT key milliseconds-timestamp

* 功能:
* 和EXPIREAT键类似,设置键在Unix时间戳指定为毫秒到期
* 返回值:
* 语法:
* 例子:

### 3.8 KEYS pattern

* 功能:
* Redis KEYS命令用于搜索与键的匹配模式
* 返回值:
* 键匹配模式的列表(数组)
* 语法:

|  |
| --- |
| redis 127.0.0.1:6379> KEYS PATTERN |

* 例子:

首先,在Redis创建一个键,并设置一定的值.

现在Redis带有键搜索

### 3.9 MOVE key db

* 功能:
* 返回值:
* 语法:
* 例子:

### 3.10 PERSIST key

* 功能:
* 返回值:
* 语法:
* 例子:

### 3.11 PTTL key

* 功能:
* 返回值:
* 语法:
* 例子:

### 3.12 TTL key

* 功能:
* 返回值:
* 语法:
* 例子:

### 3.13 RANDOMKEY

* 功能:
* 返回值:
* 语法:
* 例子:

### 3.14 RENAME key newkey

* 功能:
* 返回值:
* 语法:
* 例子:

### 3.15 RENAMENX key newkey

* 功能:
* 返回值:
* 语法:
* 例子:

### 3.16 TYPE key

* 功能:
* 返回值:
* 语法:
* 例子:

# Memcached文档