Redis

1. 介绍

Redis是一个非关系型数据库。NoSQL数据库。

Redis是一个开源的、使用C语言编写的、遵循BSD协议的、高性能的、可基于内存、亦可持久化的、Key-value 数据库。

- 开源
- C语言
- BSD

  > 这个是一个给予开发者很大自由的一个开源协议。Berkeley Software Distribution，全称叫做伯克利软件发行版

- 高性能
- 可基于内存

  > 数据存储在内存中，读写都是直接读写内存

- 亦可持久化

  > 可以把内存中的数据持久化到磁盘，保存到文件中

- key-value

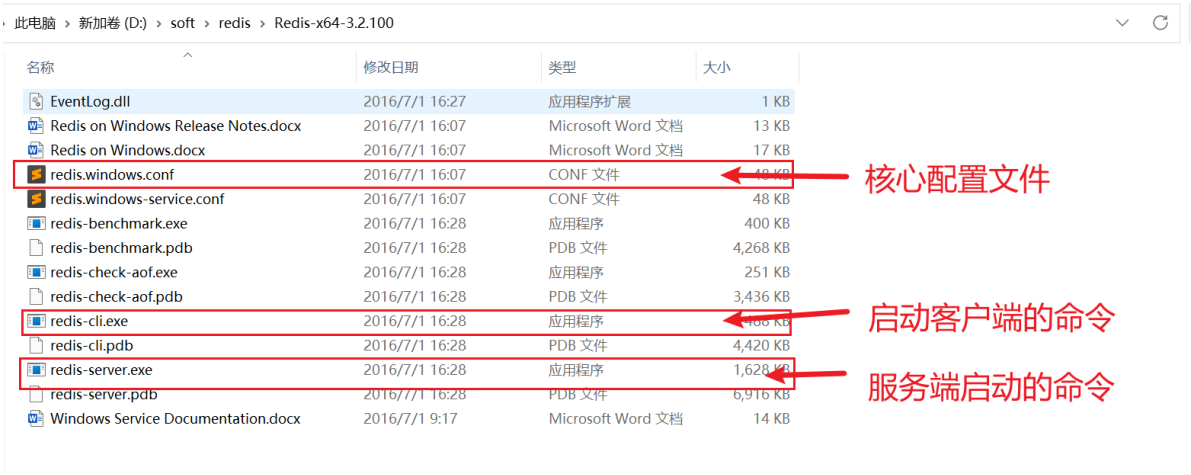
  > Redis像一个Map一样，是以键值对的方式来存储数据的。

2. 安装

2.1 下载

windows版本Redis的下载

2.2 解压



2.3 启动

> 建议不用配置Redis的环境变量

- 启动服务端

  不推荐直接双击启动，建议通过命令来启动

- 启动客户端

> 默认是没有设置密码的

3. 配置

3.1 常规配置

```
# 表示Redis是否已守护进程的方式打开，windows上暂时不支持
daemonize no

# 超时时间，单位是秒，0表示不关闭
timeout 300

# 监听的端口号，一般不修改 Merz(6379)
port 6379
```

```
# 绑定的主机地址，表示Redis接收来自哪些ip的连接
# 只能接收本地的客户端来连接
bind 127.0.0.1

# 表示接收所有的ip来访问连接
# bind 0.0.0.0

# 日志级别 （DEBUG | verbose | notice | WARNING）
loglevel notice

# 数据库的数量,编号从0开始，可以通过select index 来选择使用哪个数据库
databases 16

# 密码设置
requirepass cskaoyan
# 设置了密码之后，可以通过 auth password 来认证密码
```

3.2 持久化配置

Redis的持久化是指Redis可以把内存中的数据持久化保存到磁盘。

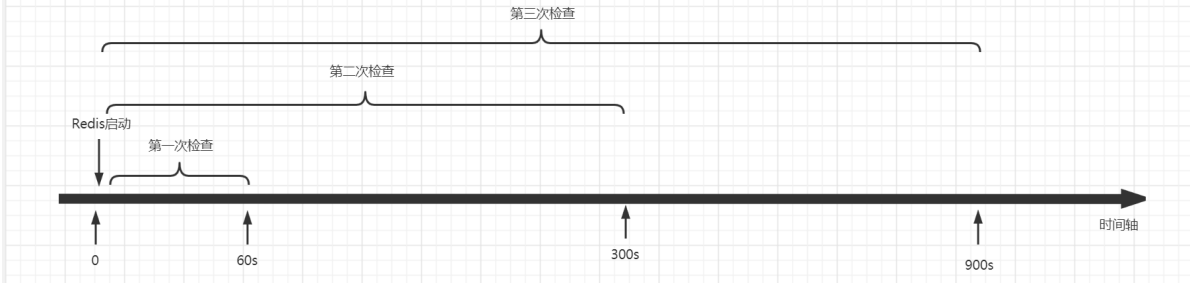
Redis的持久化有两种方式：

- RDB
- AOF

RDB

RDB是指Redis通过**内存快照**的方式把内存中完整的数据保存到磁盘。保存的是某一个时间点内存中完整的数据。RDB是Redis默认的持久化策略，默认是开启的，用户没有办法关闭。

```
# 保存的文件的路径
dir D:\tmp

# 文件的名字
dbfilename dump.rdb

# 触发策略
save 900 1
save 300 10
save 60 10000
```



> RDB的特点：
>
> 1. 保存数据比较快，还原数据也比较快

> 2. 保存的是一个完整的数据
> 3. RDB可能会丢失上一次持久化之后所有写入的数据

AOF

Append only File. AOF这种持久化的方式是通过追加日志文件来持久化的。AOF一旦开启后，Redis会把所有的写操作相关的命令保存到日志文件中，通过不断地往日志文件中追加写入的命令来持久化。

- 保存的是写入的命令
- 恢复数据的时候把文件中所有的命令再执行一次就恢复了数据库中的内容

AOF默认是关闭的，可以通过配置来开启。

```
# 总开关
appendonly yes

# 持久化文件的保存路径，和RDB是同一个配置
dir D:\tmp

# 文件的名字
appendfilename "appendonly.aof"

# 保存命令的策略
# 表示每收到一条写入的命令就同步一次,很安全，但是效率很低
# appendfsync always
# 每秒写入一次，相对来说比较安全，效率也还ok
appendfsync everysec
#让操作系统决定什么时候去把写入的命令同步的日子文件中
# appendfsync no
```

> AOF的特点：
>
> 1. 理论上来说 always可以做到不丢失数据，但是一般不使用这个配置
> 2. AOF保存数据比较慢，还原数据更慢
> 3. Redis运行时间久了之后，AOF文件会比较大，占用大量的磁盘空间

> 在公司中，一般两种持久化的策略都会开启，都开启了之后，两者并不影响各自的工作；当都开启的时候，恢复数据的时候，默认优先从AOF来恢复数据。

4. 命令

文档地址

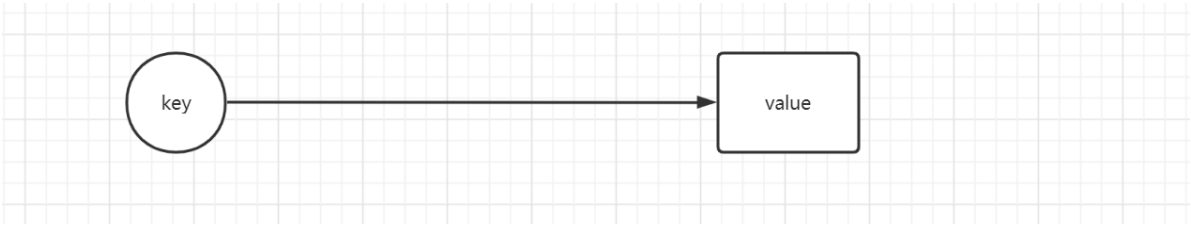
4.0 基本命令

```
# 选择数据库
select index

# 认证密码
auth password

# 查找符合特定模式的key
keys [pattern]  | keys * （查找所有的key）
```

```
# 删除当前数据库中所有的数据
flushdb

# 删除所有数据库中的数据
flushall
```

> Redis支持五种不同的数据结构，我们按照不同的数据结构来介绍不同的命令

4.1 string



这个value可以是普通的字符、也可以是整数、小数等等。

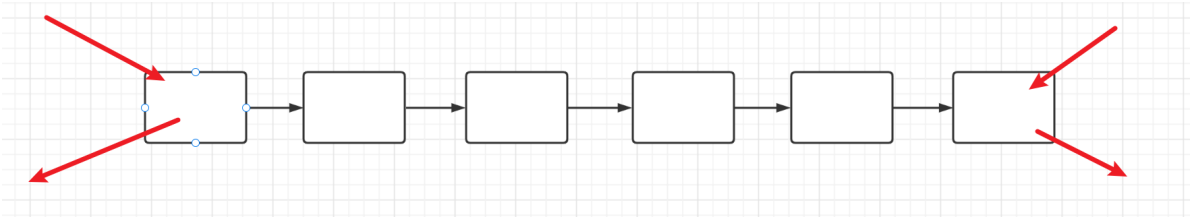
```
# 设置一个键值对的值
set key value

# 获取键值对的值
get key

# 批量设值
mset key1 value1 key2 value2 ...

# 批量获取值
mget key1 key2 ...

# 值给指定的key对应的value + 1，要求value是一个整数
incr key

# 给指定的key对应的value + increment
incrby key increment

# 设置一个key value，并且指定过期时间
setex key seconds value

# nx = not exists,表示不存在当前这个key的时候再去设值，不会覆盖原来的值
setnx key value
```

> 应用场景：
>
> 1. 可以去统计网站的访问量，PV（page view）
> 2. 存储一些普通的信息
> 3. 统计游戏的在线人数

4.2 list

双向链表。支持从链表的两端推入或者是弹出元素。

- 有序
- 可重复

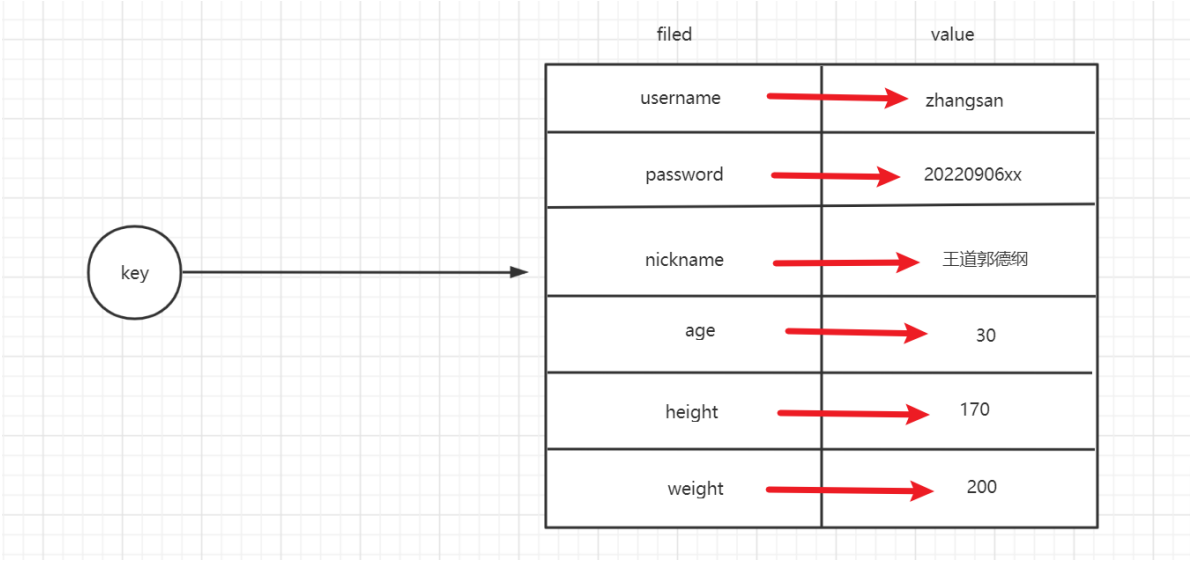


```
# 向list中添加元素,从左边推入
lpush key [value1 ...]

# 向list中添加元素,从右边推入
rpush key [value1 ...]

# 弹出元素,从左边弹出
lpop key

# 弹出元素,从右边弹出
rpop key

# 查看当前list的长度
llen key

# 查看某一个范围内的所有的元素
# start、stop表示元素的下标，最左边的元素下边为0，依此递增
lrange key start stop

# 在某个元素之前或者是之后插入一个元素
linsert key  BEFORE | AFTER pivot value
#如果指定的值有重复的，那么找第一个


# 获取指定的下标的值
lindex key index

# 当list存在的时候就去设置，如果list不存在，那么就不设值
lpushx key value

# 删除指定元素的前几个
lrem key count value

# 设置指定下标元素的值
lset key index value
```

> 应用场景：
>
> 1. 用来当做消息队列，A去生产消息，放入队列中，B从队列中获取消息，可以按照消息的生产顺序来消费消息
> 2. 最近消息列表（动态列表）

4.3 hash

hash这种数据结构就类似于二维表。



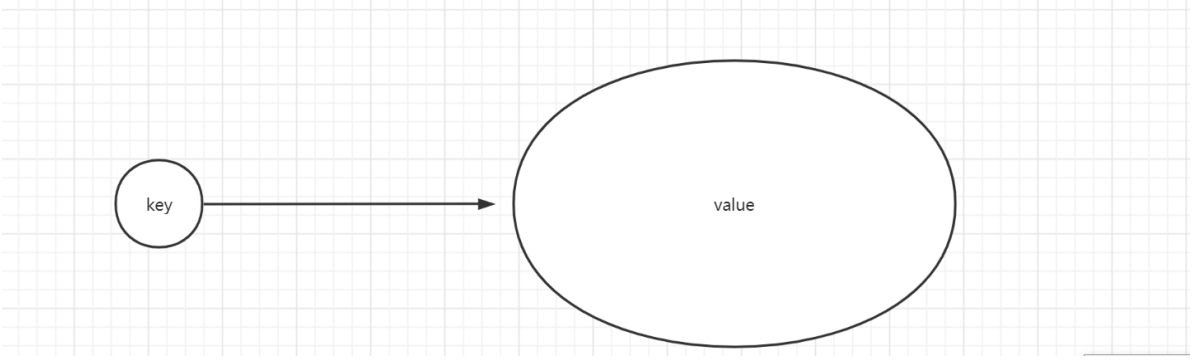

```
# 设置二维表中的一个键值对
hset key field value

# 获取二维表中指定field的值
hget key field

# 设置多个键值对
hmset key f1 v1 f2 v2 ...

# 获取多个键值对
hmget key f1 f2 ...

# 查看二维表中键值对的个数
hlen key

# 获取所有的键值对
hgetall key

# 获取二维表中所有的field
hkeys key

# 获取二维表中所有的value
hvals key

# 判断某个field是否在二维表中存在
hexists key field

# 给指定的二维表中的field对应的value增加指定的数值
hincrby key field increment

# 设置一个键值对，不会覆盖原来的值
# 假如field存在，那么不设值，假如field不存在，那么设值；假如key（二维表）都不存在，那么也会设值
hsetnx key field value
```

> 应用场景：
>
> 1. hash这种数据结构天然的适合存储Java对象，这个Java对象的名字就是 二维表的名字（key），成员变量名就是field，成员变量的值就是value，例如登录之后存储用户的相关信息。

4.4 set

无序的集合。Redis中 set这种数据结构最大的特点是可以很方便的去求set与set之间的交集、并集、差集。

- 无序
- 不可重复



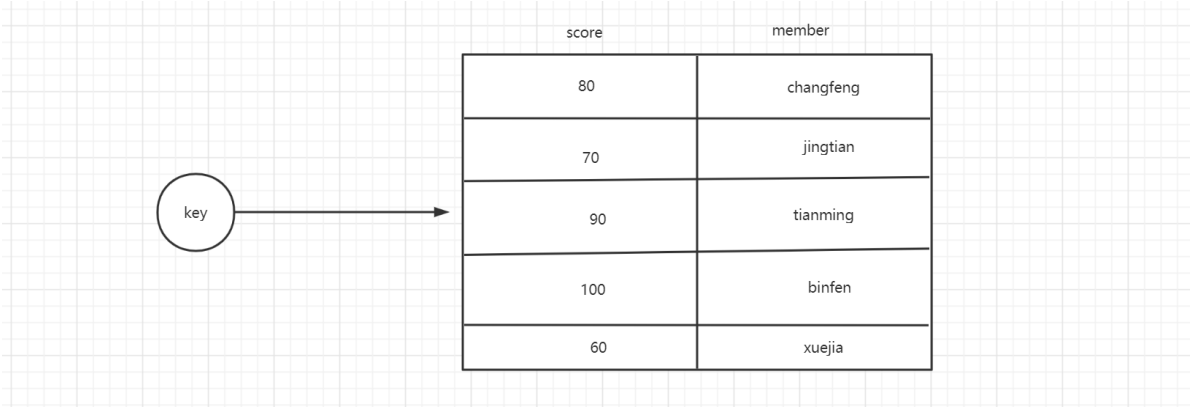

```
# 往无序集合中添加元素
sadd key value1 value2 ...

# 查询无序集合中所有的元素
smembers key

# 判断某个元素是否在集合中
sismembers key member

# 查询无序集合中元素的数量
scard key

# 从无序集合中随机取出一个元素(删除)
spop key

# 从无序集合中随机取出一个元素（不删除）
srandmember key

# 交集
sinter key1 key2 ...
# 求交集并保存
sinterstore destination key1 key2 ...

# 并集
sunion key1 key2 ...
# 求并集并保存
sunionstore destination key1 key2 ...

# 差集
sdiff key1 key2 ...
# 求差集并保存
```

```
sdiffstore destination key1 key2 ...

# 移动元素
smove source destination member

# 删除元素
srem key member1 member2 ...
```

> 1. 利用随机性，抽签、抽奖
> 2. 求共同的好友
> 3. 好友推荐

4.5 sortedset

有序集合。和hash有点类似。



> 最大的特点：可以很方便的取出任意排名区间内的成员、也可以很方便的取出任意分数区间内的成员。

```
# 往有序集合中添加成员及其对应的分数
zadd key score1 member1 score2 member2 ...

# 返回有序集合中member的个数
zcard key

# 获取指定member的分数
zscore key member

# 统计某个分数区间内成员的数量（闭区间）
zcount key min max

# 给指定的member增加分数
zincrby key increment member

# 查询指定排名区间内的成员(按照分数的升序来排名)
zrange key start stop [withscores]

# 查询指定排名区间内的成员(按照分数的降序来排名)
zrevrange key start stop [withscores]

# 查询指定分数区间内的成员(按照分数从低到高排序)
zrangebyscore key min max

# 查询指定分数区间内的成员(按照分数从高到低排序)
```

```
zrevrangebyscore key  max min

# 查询指定成员的排名（分数是从低倒高排序）
zrank key member

# 查询指定成员的排名（分数是从低倒高排序）
zrevrank key member



# 删除指定的成员
zrem

# 删除指定排名区间内的成员
zremrangebyrank

# 删除指定分数区间内的成员
zremrangebyscore
```

> 应用场景：
>
> 1. 积分排行榜
> 2. 游戏战力排行榜

5. 客户端

5.1 命令行客户端

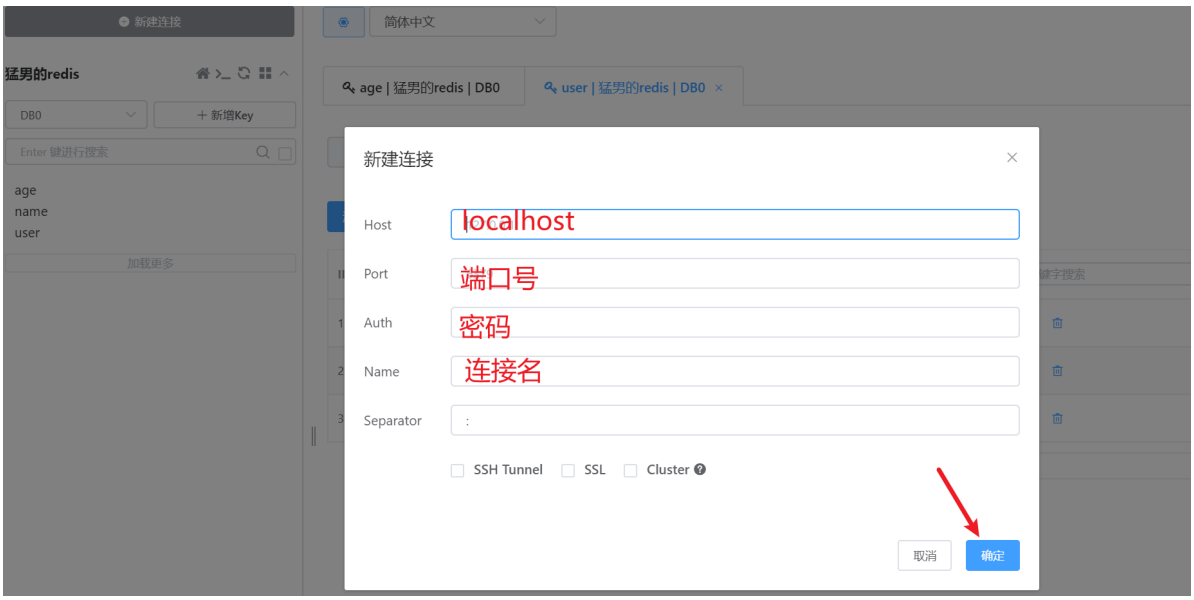
`redis-cli.exe`

这是以后工作中比较常用的

5.2 图形化界面工具

Another Redis Desktop Manager

[官方下载地址]



5.3 Java客户端

Redisson | Jedis

Jedis

Java for Redis，是业内使用的最多的一个Java客户端。Jedis上手容易，使用简单。

> Jedis最大的特点是这个客户端中提供的API和redis原生命令是保持一致的。

如何使用呢?

- 导包

```
<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>2.9.0</version>
</dependency>
```

- 配置

  Jedis目前不需要一些额外的配置。

- 使用

  > Jedis中所有的API都是和命令一致的

```
public static void main1(String[] args) {


    Jedis jedis = new Jedis("localhost",6379);

    // 密码设置
    // jedis.auth("cskaoyan");


    //        jedis.set("name","张三");

    String name = jedis.get("name");

    System.out.println("name:" + name);

}
```

6. 内存淘汰策略

Redis的内存淘汰策略是指：Redis是基于内存来存储的一个数据库，内存资源是十分宝贵且有限的。那么假如当前Redis存储了大量的数据，导致内存满了，这个时候 又有新的数据需要来存储，这个时候Redis该怎么办呢?

Redis就要根据当前配置的内存淘汰策略来淘汰掉部分的旧数据，来保证写入的成功。

- volatile-lru

  从已经设置过期时间的数据集中，挑选（least recently used）最近最少使用的数据进行淘汰

- volatile-lfu

从已经设置过期时间的数据集中，挑选一段时间内使用次数最少key进行淘汰

- volatile-ttl

  从已经设置过期时间的数据集中，挑选最近将要过期的数据进行淘汰

- volatile-random

  从已经设置过期时间的数据集中，随机选择数据进行淘汰

- allkeys-lru

  从所有的数据集中，挑选（least recently used）最近最少使用的数据进行淘汰

- allkeys-lfu

  从所有的数据集中，挑选一段时间内使用次数最少key进行淘汰

- allkeys-random

  从所有的数据集中，随机选择数据进行淘汰

- no-evication

  禁止驱逐数据。如果内存满了，那么这个时候有新的数据来写入，不会淘汰旧数据，会直接拒绝写入

> 在以后的公司中，我们一般会选择哪种内存淘汰策略呢？
>
> - 从合理性的角度来说，volatile-lru是最合理的策略
> - 从效率的角度来说，allkeys-random效率最高
>
> 大多数情况下都会更加偏重于淘汰的效率，选择allkeys-random这个内存淘汰策略。
>
> 在以后的工作中，应该尽量的让Redis少触发或者是不主动触发内存淘汰，那么应该怎么做呢？
>
> - 保证机器有一个比较大的内存
> - 主动设置key-value的过期时间，让这些过期的数据自动被清空，少触发内存淘汰

> 总结：什么样的数据适合存储在Redis呢？
>
> 1. 用户不敏感的数据
> 2. 访问量比较大，并发比较高的数据

缓存雪崩 | 缓存击穿