# Redis安装

官网

https://redis.io/

redis是c编写的所以安装运行，需要有gcc的环境

建议在linux中安装redis

使用docker安装redis

$ docker search redis //查找镜像

$ docker pull redis //获取镜像

**追踪**一些运行数据，及缓存数据，分布式session共享

## Redis启动

$ docker run - -name myredis -p 6379:6379 -v $PWD/data:/data -d docker.io/redis - -requirepass “123456”

→ requirepass : 设置密码

$ docker exec -it CONTAINERID redis-cli

# jedis

https://github.com/xetorthio/jedis

## 导入依赖包

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

<version>2.7.3</version>

<type>jar</type>

<scope>compile</scope>

</dependency>

## 测试连接

|  |
| --- |
| **public** **void** demo1() {  // 1.创建一个jedis对象，传入服务器ip地址  Jedis jedis = **new** Jedis("139.199.167.152",6379);  // 如果设置了认证，就需要认证一下  jedis.auth("123456");  // 2.保存数据  jedis.set("name","hello world!!");  // 3.获取数据  String value = jedis.get("name");  System.***out***.println(value);  // 4.释放资源  jedis.close();  } |
| // 使用连接池  **public** **void** demo2() {  // 配置连接池的属性  JedisPoolConfig config = **new** JedisPoolConfig();  config.setMaxTotal(30);  config.setMaxIdle(10);  // 创建连接池对象  JedisPool pool = **new** JedisPool(config, "192.168.3.12", 6379);  // 创建Jedis对象  Jedis jedis = **null**;  **try** {  jedis = pool.getResource();  // 存入数据  jedis.set("name", "hello world");  // 取出  String value = jedis.get("name");  System.***out***.println(value);  }**catch**(Exception e) {  e.printStackTrace();  }**finally** {  //关闭资源  **if** (jedis != **null**) {  jedis.close();  }  **if** (pool != **null**) {  pool.close();  }  }  } |

# Redis的数据类型

字符串String 哈希hash

字符串列表list 字符串集合set

有序字符串集合sorted set

Key定义的注意点：

* 不要过长
* 不要过短
* 要有统一的命令规范

## 字符串String

set key value | get key | getset key | incr key | decr key | incrby key number |

decrby key number | append key value |

## 哈希hash

hset key v\_key v\_value | hget key v\_key | hmget | hgetall key |

hmset //设置多个v\_key和v\_value

hdel key v\_key v\_key ... //删除多个值

del key //删除整个集合

hincrby key v\_key number

hexists key v\_key //查看是否存在

hlen key //查看属性的长度

hkeys key //查看key中的所有的属性

hvals key //查看key中的所有的值

## redis教程

https://www.w3cschool.cn/redis/redis-lists.html

# 相关特性

## 多数据库

一个redis对象有16个数据库，默认使用index=0的数据库；

> select 1 //选择数据库

> move key index //把当前数据库中的key的数据移动到index数据库中

## 事务

multi // 开启事务 begin

exec // 提交 commit

discard // 回滚 reback

# redis持久化

两种持久化方式：RDB方式、AOF方式

* RDB持久化(redis默认)：

在指定的时间间隔内将内存中的数据及快照写入到磁盘

* AOF持久化：

以日志的方式记录redis的所有操作，在redis服务器启动的时候读取此文件，重新构建数据库，保证数据库中的数据是完整的

* 无持久化

可以通过配置禁用持久化，这时只是用作缓存数据库

* 同时使用RDB与AOF持久化

## RDB持久化

劣势：子线程进行持久化，当发生灾难的时候，会丢失一些数据

配置：

# vi /usr/local/redis/redis.conf

配置文件中的RDB配置：

save 900 1 //每900s有1个key发生变化，备份一次

save 300 10 //每300s有10个key发生变化，备份一次

save 60 10000 //每60s有10000个key发生变化，备份一次

dbfilename dump.rdb //备份的文件名

dir ./ //备份的路径

## AOF持久化

使用append操作追加日志

劣势：AOF文件大于RDB文件，效率低于RDB

redis中AOF有3种同步策略：每秒同步、每修改同步、不同步

配置：

appendonly no //默认关闭AOF；设置yes为开启AOF

appendfilename “appendonly.aof”

#appendfsync always //每修改同步

appendfsync everysec //每秒同步

#appendfsync no //不同步

测试：

1. 打开AOF同步方式
2. 一些操作
3. flushall //清空数据库
4. 退出redis
5. 找到appendonly.aof文件，打开把里边最后一条命令flushall删除
6. 重新运行redis，删除的数据就又回来了