目录

[一、什么是nosql 1](#_Toc426126407)

[（1）NoSQL的基本说明： 1](#_Toc426126408)

[（2）兴起的原因： 1](#_Toc426126409)

[（3）特点： 2](#_Toc426126410)

[（4）常见的nosql产品 2](#_Toc426126411)

[二、redis的基本介绍 2](#_Toc426126412)

[三、redis的安装 3](#_Toc426126413)

[四、redis的数据类型 4](#_Toc426126414)

[五、redis的数据类型讲解相关命令 6](#_Toc426126415)

[1、string类型 6](#_Toc426126416)

[2、hashes类型 10](#_Toc426126417)

[3、lists类型及操作 13](#_Toc426126418)

[4、sets类型及操作 16](#_Toc426126419)

[5、sorted sets类型及操作 20](#_Toc426126420)

[6、redis常用命令 24](#_Toc426126421)

[六、redis的安全性。 29](#_Toc426126422)

[七、持久化机制 29](#_Toc426126423)

[九、php支持redis 31](#_Toc426126424)

[十、案例完成用户注册系统。 33](#_Toc426126425)

[十一、redis常用命令 38](#_Toc426126426)

# 一、什么是nosql

## （1）NoSQL的基本说明：

NoSQL(NoSQL = Not Only SQL),意味反sql运动，是一项全新的数据库革命性运动，早期就有人提出，发展至2009年趋势越发高涨。它指的是**非关系型**的数据库。

## （2）兴起的原因：

随着互联网web2.0网站的兴起，传统的关系数据库在应付web2.0网站，特别是**超大规模**和**高并发**的sns类型的web2.o纯动态网站已经显得力不从心，暴露了很多难以克服的 问题，而非关系型的数据库则由于本身的特点得到了非常迅速的发展。

## （3）特点：

**优点：**

**高并发读写的性能**

**大数据量的扩展（分布式存储）**

配置简单

灵活、高效的操作与数据模型

低廉的成本

**不足之处：**

没有统一的标准

没有正式的官方支持

各种产品还不算成熟

## （4）常见的nosql产品

新浪微博 Redis

Google Bigtable

Amazon SimpleDB

淘宝数据平台 Tair

优酷视频 MongoDB

飞信空间 HandlerSocket

视觉中国网站 MongoDB

# 二、redis的基本介绍

redis是一个开源的，先进的key-value存储。它通常被称为数据结构服务器，因为键可以包含**字符串**，**哈希**，**链表**，**集合**和**有序集合**。

（1）架构方式：c/s 客户端和服务器

（2）memcache存储数据的位置：内存中。

redis：可以存储到内存中，可以把数据同步 到硬盘中，达到数据的存储持久化。

（3）memcache存储数据类型的格式：键值对。没有行和列的概念

redis：数据存储是，键值对存储，只不过存储的值的类型比较丰富，有五种类型，string(字符串),hash(哈希)，list(链表),set(集合)，zset(有序集合)

基本介绍：

（1）Redis是一个**Key-Value**存储系统。

（2）它支持存储的**value类型很多**，包括**string(字符串)**、**list(链表)**、**set(集合)、Zset(有序集合)**。这些数据类型都支持push/pop、add/remove及取交集和并集及更丰富的操作、Redis支持各种不同方式的排序。

（3）为了保证效率数据都是**缓存在内存中**，它也可以周**期性**的把更新的数据**写入磁盘**或者把修改操作写入追加的记录文件。

（4）提供的API语言包括：C、C++、C#、Clojure Common Lisp Erlang Haskell Java Javascript Lua Objective-C Perl **PHP** Python Ruby Scala Go Tcl

# 三、redis的安装

（1）下载安装包：

<http://redis.googlecode.com/files/redis-2.4.17.tar.gz>

（2）编译程序：

tar zxvf redis-2.4.17.tar.gz

cd redis-2.4.17

make

cd src && make install

（3）为了便于管理，新建redis的管理目录



（4）把配置文件移动到新建的etc目录下面（配置文件在redis解压目录中）。



（5）把解压目录下面的src目录下面的一些文件移动到新建的bin目录下

mv mkreleasehdr.sh redis-benchmark redis-check-aof redis-check-dump redis-cli redis-server /usr/local/redis/bin



（6）启动服务，通过bin目录下面的redis-server

启动格式：redis-server 指定的配置文件



启动后，会占用当前的会话窗口，我们应该让他在后台启动，因此要修改配置文件。

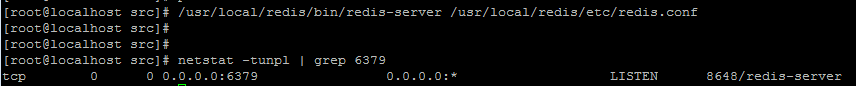
（7）修改配置文件，让他从后台启动。打开配置文件/etc/redis.conf,把

daemonize如果需要在后台运行，把该项改为yes

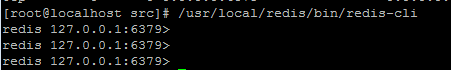


改为yes

（8）关闭redis的服务重新启动。



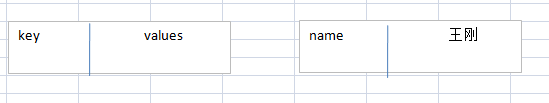
（9）客户端连接。



## 四、redis的数据类型

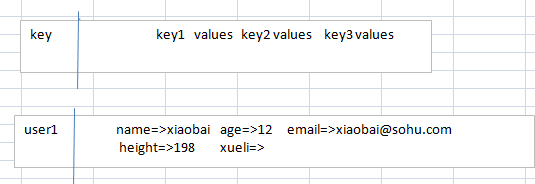
string(字符串)：类似于memcache里面的数据存储方式

key values



hash(哈希)：类似于关系型数据库里面的行，

也是键值对，值里面又有键值对。

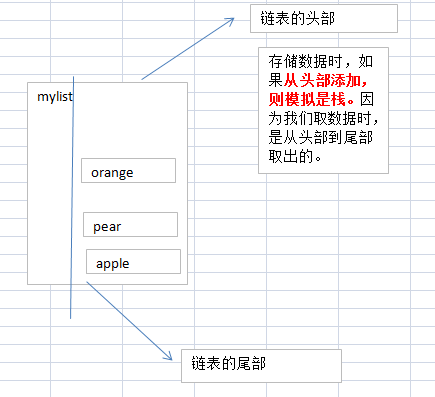


list(链表)：可以模拟队列和栈。数据存储是有顺序的。

队列：先进先出。

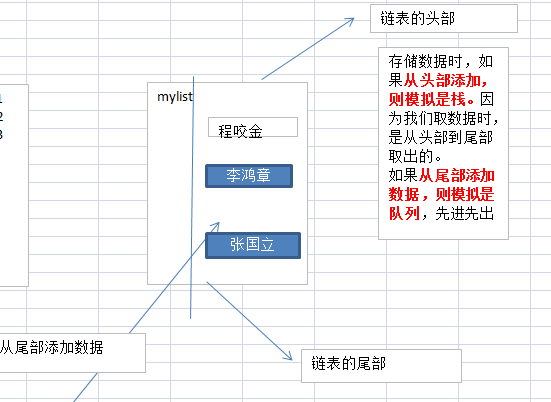
栈：先进后出。

模拟栈



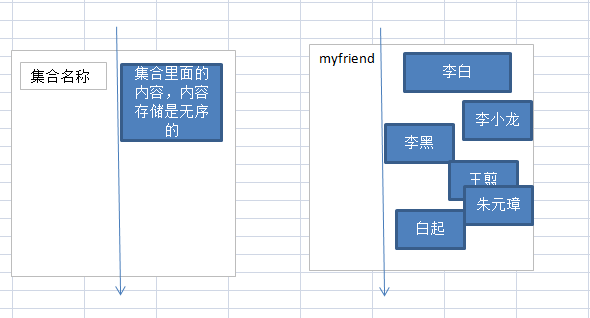
从头部添加

模拟队列：

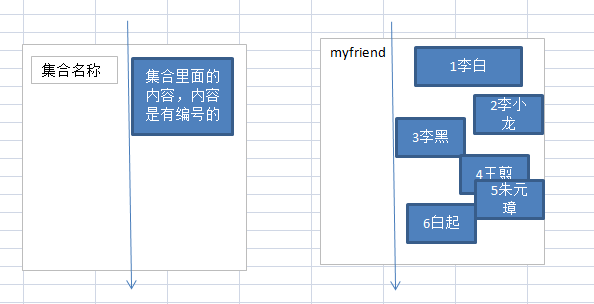


set(集合)

集合的示意图：



zset(有序集合) 集合里面的内容是有序的。



# 五、redis的数据类型讲解相关命令

## 1、string类型

String是最简单的类型，一个 key对应一个Value，String类型是二进制安全的。Redis的 string可以包含任何数据，比如jpg图片或者序列化的对象。

（1）set

设置键，值

语法：set 键名称 值

例如：我们添加一个name=“hanguo”的键值对。

重新设置则直接覆盖。



（2）**get**

获取key对应的string值，如果key不存在返回 nil,

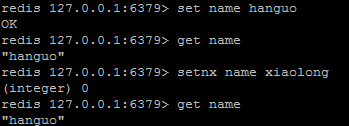
语法：get 键值



（3）**Setnx**

设置键时，先判断一下该键是否存在，如果key已经存在，返回0,nx是not exist的意思。

若键已经存在，则设置不成功，返回0

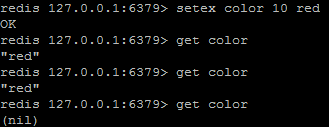


（4）**setex**

设置key对应的值为string类型的value,并指定此键值对应的有效期。

语法：setex 名称 有效期 值

例如：我们添加一个color = red的键值对，并指定有效期为10秒。

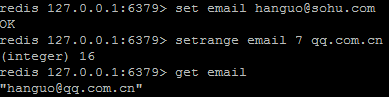


**（5）setrange**

替换字符串中某些字符

语法：setrange 键名称 开始替换的序号 替换为的内容

例如：我们希望将hanguo的sohu邮箱替换为qq.com邮箱

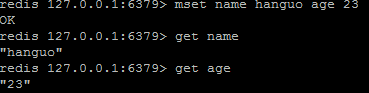


（6）**mset**

一次设置多个key的值，成功返回ok表示所有的值都设置了，失败返回0表示没有任何值被设置。

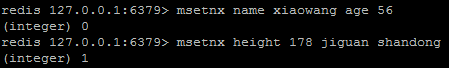
语法：

mset 名称1 值1 名称2 值2



（7）**msetnx**

一次设置多个key的值，成功返回ok表示所有的值都设置了，失败返回0表示没有任何值被设置，但是不会覆盖已经存在的key。



（8）**getset**

设置key的值，并返回key的旧值。（设置新值，获取旧值。）



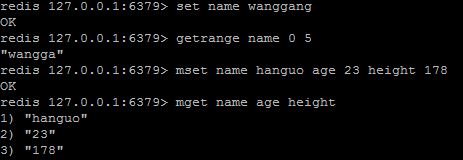
（9）**getrange**

获取key的value值的范围内的子字符串

getrange name 0 5

（10）**mget**

一次获取多个key的值，如果对应key不存在则对应返回nil。

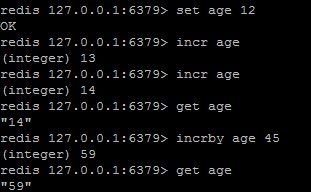


（11）**incr**

对key的值做加加操作，并返回新的值。

**incrby**

同incr类似，加指定值，key不存在时候会设置key,并认为原来的value是0。

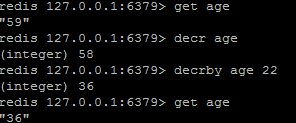


（12）**decr**

对key的值做减减操作。

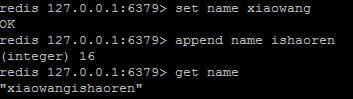
**decrby**

同decr类似，减指定值



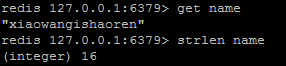
（13）**append**

给指定key的字符串追加value,返回新字符串值的长度。



（14）**strlen**

取指定key的value值的长度。



## 2、hashes类型

Redis hash是一个string类型的field和value的映射表。它的添加、删除操作都是0（1）（平均）。hash特别适合用于存储对象。相较于将对象的每个字段存成单个string类型。将一个对象存储在hash类型中会占用更少的内存，并且可以更方便的存取整个对象。

（1）hset

设置hash field为指定值，如果 key不存在，则先创建。

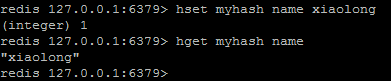
语法：

hset 哈希名称 字段名称 值



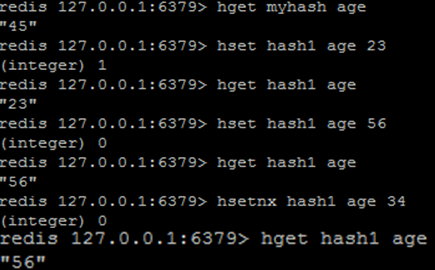
（2）hget

取出hash field的值。语法：hget 哈希名称 字段名称



（3）hsetnx

设置hash field为指定值，如果key不存在，则先创建，如果存在则返回0。



（4）hmset

同时设置hash的多个field

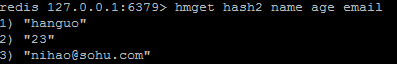
语法：hmset 哈希名称 field1 value1 field2 value2



（5）hmget

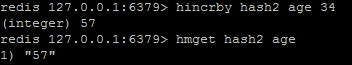
获取全部指定的hash field。

语法：hmget 哈希名称 field1 field2



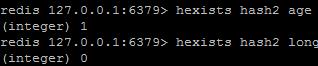
（6）hincrby

指定的 hash field加上给定的值。



（7）hexists

测试指定的 field是否存在。



（8）hlen

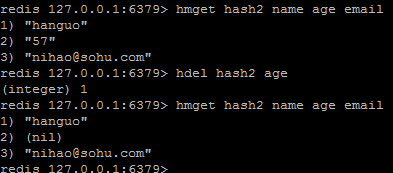
返回指定hash的field数量。



（9）hdel

删除指定hash的field

语法：hdel 哈希名 field



（10）hkeys

返回hash的所有field



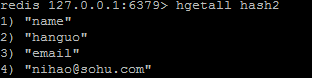
（11）hvals

返回hash的所有 value。



（12）hgetall

获取某个hash中全部的field及value



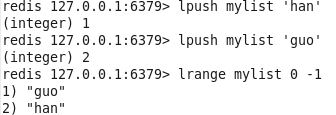
## 3、lists类型及操作

list是一个链表结构，主要功能是push、pop、获取一个范围的所有值等等，操作中key 理解为链表的名字。redis的list类型其实就是一个每个子元素都是string 类型的双向链表。我们可以通过push、pop操作从链表的头部或者尾部添加删除元素，这样list即可以作为栈，又可以作为队列。

（1）lpush

在key对应list的头部添加字符串元素。

语法：lpush 链表名称 值内容



（2）lrange

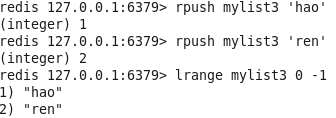
获取链表里面的值，语法：lrange 链表名称 0 -1

注意0 和 -1 表示取值范围，从头部到尾部。

（3）rpush

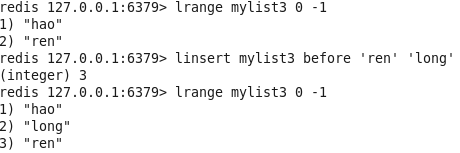
在key对应list的尾部添加字符串元素。

语法：rpush 链表名称 值内容



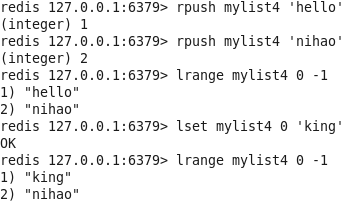
（3）linsert

在key对应list的特定位置前或后添加字符串。



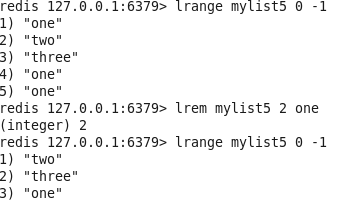
（4）lset

设置list中指定下标的元素值。注：下标从0开始计算



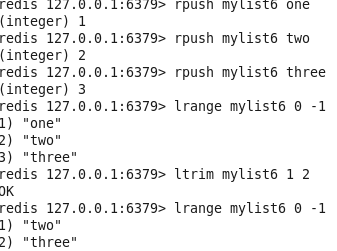
（5）lrem

从key对应list中删除n个和value相同的元素。（n<0从尾删除，n=0全部删除）



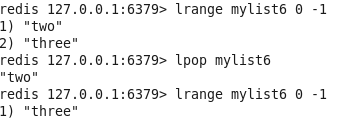
（6）**ltrim**

**保留指定key的值范围内的数据。**



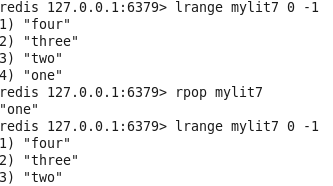
**（7）lpop**

**从list的头部删除元素，并返回删除元素。**



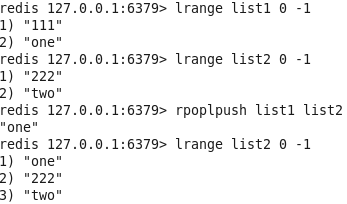
（8）rpop

从 list的尾部删除元素，并返回删除元素。



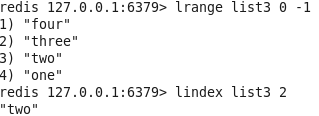
（9）**rpoplpush**

**从第一个list的尾部移除元素并添加到第二个list的头部。**

****

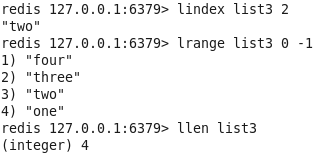
（10）lindex

返回名称为key的list中 index位置的元素。



（11）llen

返回key对应list的长度。



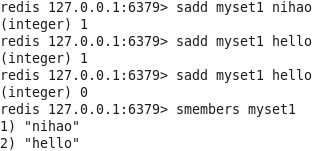
## 4、sets类型及操作

set是集合，它是string类型的无序集合。set是通过hash table实现的、添加、删除和查找的复杂度都是0(1)。对集合我们可以取并集、交集、差集。通过这些操作我们可以实现sns中的好友推荐和blog的tag功能。

（1）sadd

向名称为key 的set中添加元素。

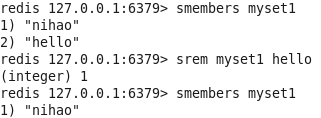
语法：sadd 集合名 元素



（2）smembers 获取集合中内容，语法：smembers 集合名称

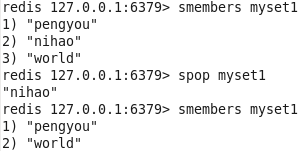
（3）srem

删除名称为key的set中的元素.



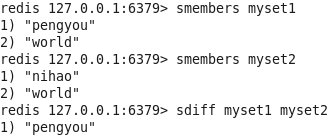
（4）spop

随机返回并删除名称为key的 set中一个元素。



（5）sdiff

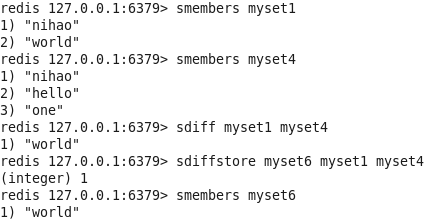
返回所有给定key与第一个key的差集。



（6）sdiffstore

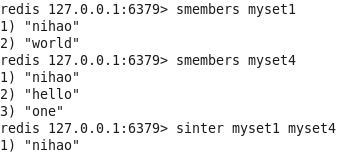
返回所有给定key与第一个key的差集，并将结果存为另一个key。

语法：sdiffstore 新的集合 集合1 集合2



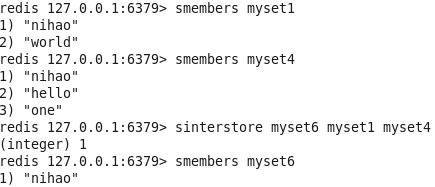
（7）sinter

返回所有给定key的交集.



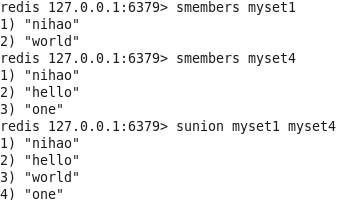
（8）sinterstore

返回所有给定key的交集，并将结果存为另一个key



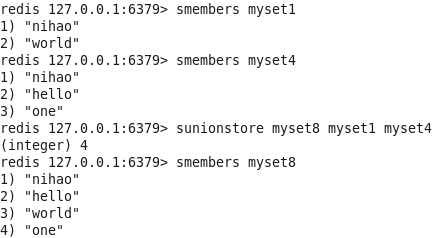
（9）sunion

返回所有给定key的并集



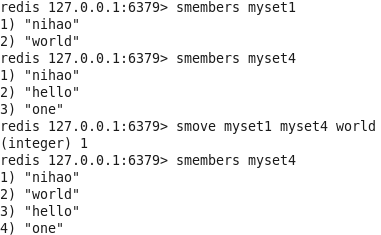
（10）sunionstore

返回所有给定key的并集。



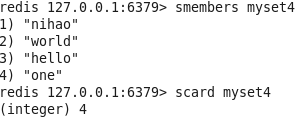
（11）smove

从第一个key对应的set中移除member并添加到第二个对应的set中。



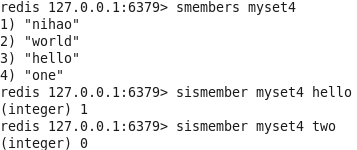
（12）scard

返回名称为key的set的元素个数。



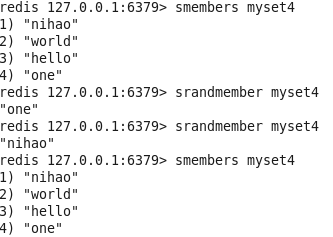
（11）sismember

测试member是否是名称为 key的set的元素



（12）srandmember

随机返回名称为key的set的一个元素，但不删除元素



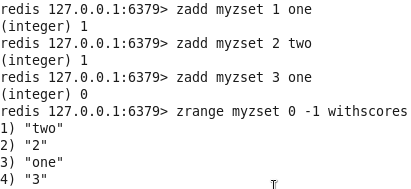
## 5、sorted sets类型及操作

sorted set是set的一个升级版本，他在set的基础上增加了一个顺序属性，这一属性在添加修改元素的时候可以指定，每次指定后，zset会自动重新按新的值调整顺序。可以理解为有两列的mysql表，一列存value,一列存顺序。操作中的key理解为zset的名字。

（1）zadd

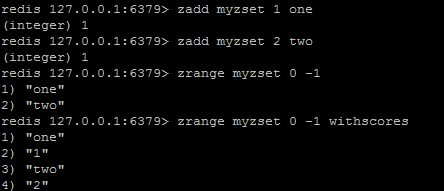
向名称为key的zset中添加元素。如果该元素存在，则更新其顺序。

语法：zadd 集合名 序号 内容



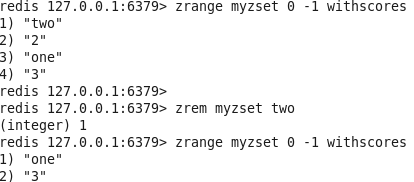
（2）zrange

获取有序集合中的内容



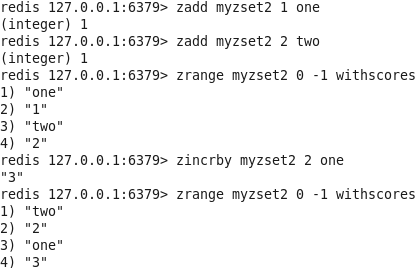
（3）zrem

删除名称为key的zset中的元素member。



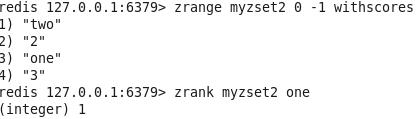
（4）zincrby

如果在名称为key的zset中已经存在元素member,则该元素的score增加increment否则向该集合中添加该元素，其score的值为increment。



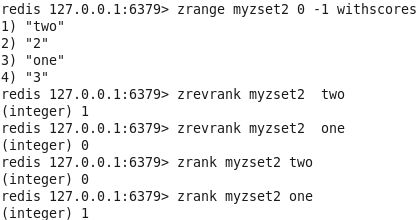
（5）zrank

返回名称为key的zset中member元素的排名（按score从小到大排序）即下标。



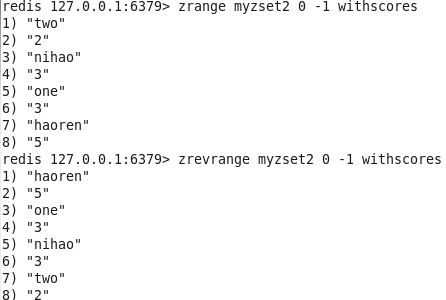
（6）zrevrank

返回名称为key的zset中member元素的排名（按score从大到小排序）即下标。



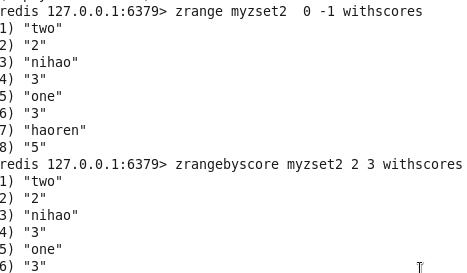
（7）zrevrange

返回名称为key的zset(按score从大到小顺序)中的index从start到end的所有元素。



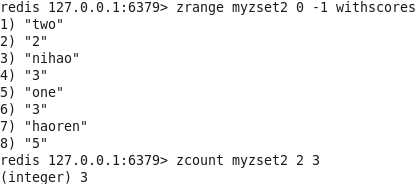
（8）zrangebyscore

返回集合中score在给定区间的元素



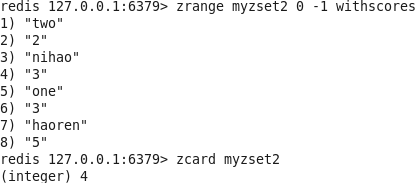
（9）zcount

返回集合中score在给定区间的数量。



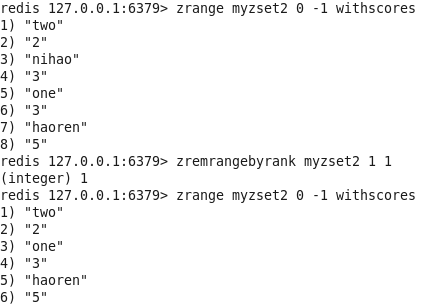
（10）zcard

返回集合中元素的个数



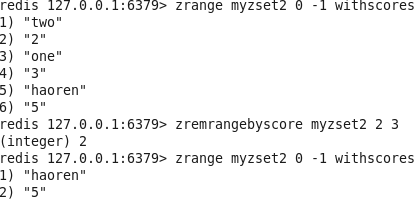
（11）zremrangebyrank

删除集合中排名在给定区间的元素。



（12）zremrangebyscore

删除集合中score在给定区间的元素。



## 6、redis常用命令

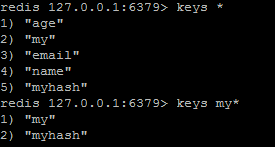
redis提供了丰富的命令对数据库和各种数据库类型进行操作，这些命令可以在linux终端使用。

1、键值相关的命令

（1）keys

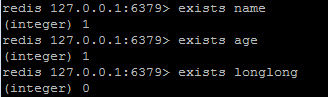
返回满足给定参数的所有key

用表达式\*,代表取出所有的key



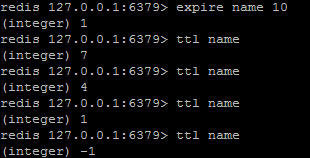
（2）exists

确认一个key是否存在。



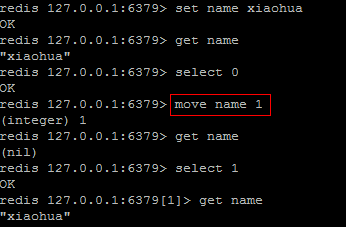
（3）**expire**

**设置一个key的过期时间**



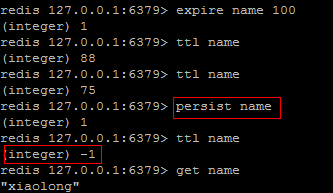
（4）move:

将当前数据库中的key转移到其它数据库中。



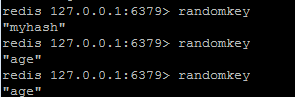
（5）persist

移除给定key的过期时间



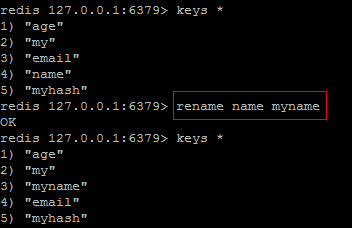
（6）randomkey

随机返回key空间的一个key

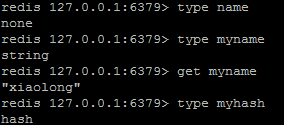


（7）rename

重命名key



（8）type:返回值的类型

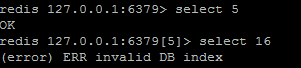


2、服务器相关命令

（1）select

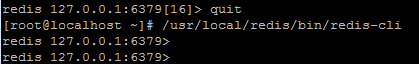
选择数据库。redis数据库编号从0-15，我们可以选择任意一个数据库来进行数据的存取。

当选择16时，报错，说明没有编号为16的这个数据库。



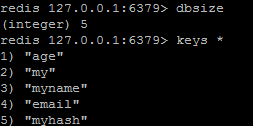
（2）quit

退出连接



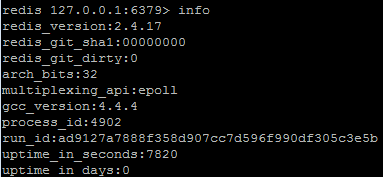
（3）dbsize

返回当前数据库中key的数目



（4）info

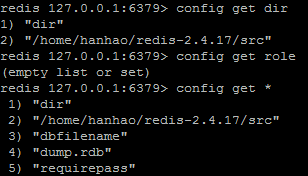
获取服务器的信息和统计。



（5）config get

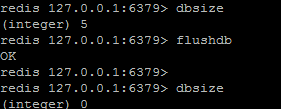
获取参数的配置。

本例中我们获取了dir这个参数配置的值，如果想获取全部参数的配置只需要执行”config get \*”即可将全部的值都显示出来



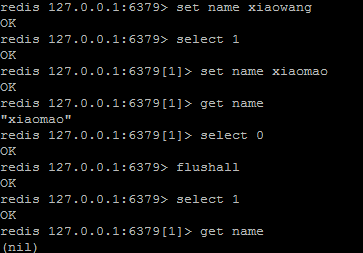
（6）flushdb

删除当前选择数据库中的所有key



（7）flushall

删除所有数据库中的所有的 key



# 六、redis的安全性。

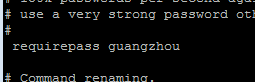
设置客户端连接后进行任何其他操作前需要使用的密码。

注意：因为 redis速度相当快，所以在一台比较好的服务器下，一个外部的用户可以在一秒钟进行150k次的密码尝试，这意味着你需要指定非常非常强大的密码来防止暴力破解。

（1）设置密码的方式，打开redis的配置文件。

#requirepass 设置的密码

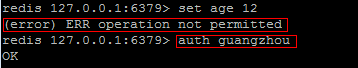
比如：requirepass guangzhou



设置完密码后，redis服务要重启。

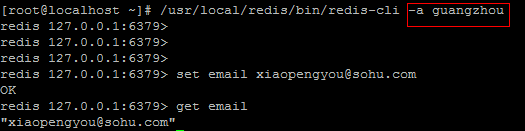
（2）设置完成密码后，客户端登录方式：

第一种方式：登录后，执行**auth 设置的密码**  后才能操作，否则不能任何操作。



第二种方式：在登录时，输入密码

/usr/local/redis/bin/redis-cli **–a 设置的密码**

****

# 七、持久化机制

**redis是一个支持持久化的内存数据库，也就是说redis需要经常将内存中的数据同步到硬盘来保证持久化。**

**redis支持两种持久化方式：**

**（1）snapshotting(快照)默认方式**

**快照是默认的持久化方式。这种方式是将内存中数据以快照的方式写入到二进制文件中，默认的文件名为dump.rdb.可以通过配置设置自动做快照持久化的方式。我们可以配置redis在n秒内如果超过m个key修改就自动做快照。**

****

**快照方式的缺点：**

**由于快照方式是在一定间隔做一次的，所以如果redis意外down掉的话，就会丢失最后一次快照后的所有修改。**

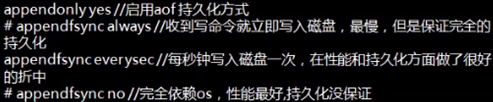
**（2）append-only file( 缩写aof)的方式**

**aof比快照方式有更好的持久化性，是由于在使用aof时，redis会将每一个收到的写命令都通过write函数追加到文件中，当redis重启时会通过重新执行文件中保存的写命令来在内存中重建整个数据库的内容。**

**aof方式**

**当然由于os会在内核中缓存write做的修改，所以可能不是立即写到磁盘上。这样aof方式的持久化也还是有可能会丢失部分修改。**

**可以通过配置文件告诉reids我们想要通过fsync函数强制os写入到磁盘的时机。**

****

**八、redis配置文件常用的项**

**daemonize如果需要在后台运行，把该项改为yes**

**pidfile:配置多个pid的地址 默认在/var/run/redis.pid**

**bind：绑定ip,设置后只接受来自该ip的请求。**

**port：监听端口，默认为6379**

**timeout：设置客户端连接时的超时时间，单位为秒。**

**logfile：配置log文件的地址。**

**databases：设置数据库的个数，默认使用的数据库为0**

**sava：设置redis进行数据库镜像的频率。**

**rdbcompression:在进行镜像备份时，是否进行压缩。**

**Dbfilename:镜像备份文件的文件名。**

**Dir:数据库镜像备份的文件放置路径。**

**Slaveof：设置数据库为其他数据库的从数据库。**

**Masterauth：主数据库连接需要的密码验证。**

**Requirepass：设置登录时需要使用的密码。**

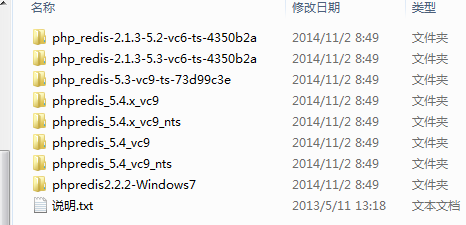
**Maxclients:限制同时连接的客户数量**

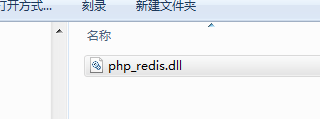
**Maxmemory：设置redis能够使用的最大内存。**

**Appendonly：开启append only模式。**

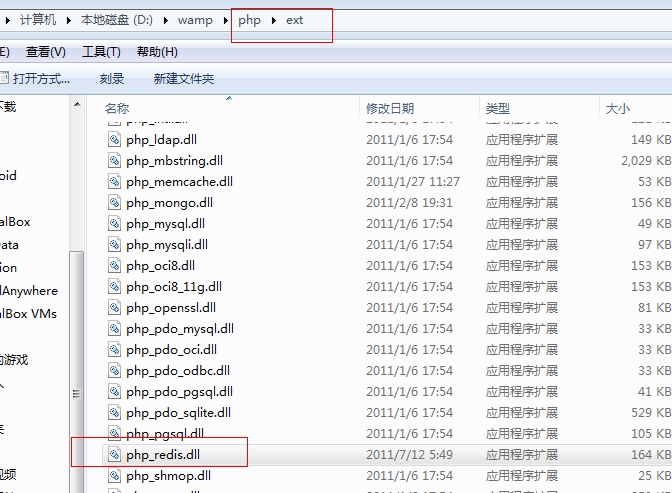
# 九、php支持redis

（1）下载扩展，在下载扩展时要注意，和php的版本要兼容。

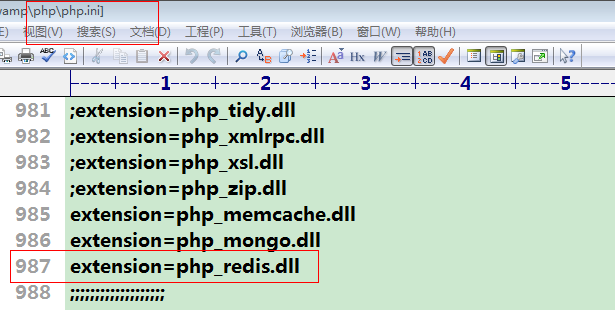
****

****

**（2）把对应的扩展拷贝到，php的安装目录ext目录下面。**

****

**（3）打开php.ini文件，引入扩展。**

****

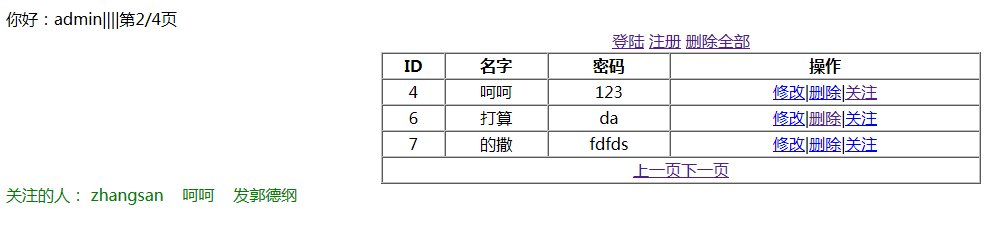
**（4）重启apache进行测试，写一个文件，运行phpinfo()函数测试；**

****

**（5）基本使用**

****

# 十、案例完成用户注册系统。



**1.index.php 主页、显示用户列表、登陆情况、用户关注的人**

<?php

include\_once 'redis.php';

session\_start();

//分页

//获得链表长度（总记录数）

$rows=$redis->lsize('uid');

//计算总页数

$count=ceil($rows/3);

//配置当前页

$page=isset($\_GET['page'])?max(1,min($\_GET['page'],$count)):1;

//起始偏移量

$s=($page-1)\*3;

//结束偏移量

$e=$page\*3-1;

//分页取出数据

$uid=$redis->lrange('uid',$s,$e);

$arr=array();

foreach($uid as $id){

//根据ID哈希中取出数据

$arr[]=$redis->hgetall('user:'.$id);

}

$\_uid=0;

if(isset($\_SESSION['uid'])){

$\_uid=$\_SESSION['uid'];

echo '你好：'.$\_SESSION['username'].'||||第'.$page.'/'.$count.'页';

}

?>

<div align='center'>

<span><a href='login.php'>登陆</a></span>

<span><a href='register.php'>注册</a></span>

<span><a href='delall.php' onclick='return confirm()'>删除全部</a></span>

<table border=1 cellspacing=0 width=600>

<tr><th>ID</th><th>名字</th><th>密码</th><th>操作</th></tr>

<?foreach($arr as $row): if($row['id']!=$\_uid){?>

<tr align=center><td><?echo $row['id']?></td><td><?echo $row['username']?></td><td><?echo $row['password']?></td><td><a href='edit.php?id=<?echo $row['id']?>'>修改</a>|<a href='del.php?id=<?echo $row['id']?>'>删除</a>|<a href='care.php?id=<?echo $row['id']?>'>关注</a></td></tr>

<?}endforeach;?>

<tr align=center><td colspan=4><a href='index.php?page=<?echo $page-1?>'>上一页</a><a href='index.php?page=<?echo $page+1?>'>下一页</a></td></tr>

</table>

</div>

<!--关注的人-->

<div style='color:green'>

<?if(isset($\_SESSION['uid'])&&$redis->smembers('care\_user:'.$\_SESSION['uid'])){

echo '关注的人：';

//取出集合中的ID

$arr=$redis->smembers('care\_user:'.$\_SESSION['uid']);

foreach($arr as $id){

//根据ID取出用户名

$row=$redis->hget('user:'.$id,'username');?>

<span><?echo $row?>&nbsp&nbsp&nbsp</span>

<?}

}?>

</div>

**2.redis.php redis类**

<?php

echo '<meta charset=utf-8>';

$redis=new redis();

//连接redis服务器

$redis->connect('192.168.23.200');

//密码

$redis->auth('admin');

?>

**3.register.php 注册、处理**

<?php

include\_once 'redis.php';

if(isset($\_POST['username'])){

$username=$\_POST['username'];

$password=$\_POST['password'];

//设置自增长ID

$id=$redis->incr('incr');

//将ID存到链表中

$redis->rpush('uid',$id);

//根据每个用户名创建字符串类型存储ID

$redis->set('user:'.$username,$id);

//将数据存到哈希中

$redis->hmset('user:'.$id,array('id'=>$id,'username'=>$username,'password'=>$password));

header('location:index.php');

}

?>

<div align='center'>

<h1>注册用户</h1>

<form action='register.php' method='post'>

名字：<input type='text' name='username'><br>

密码：<input type='password' name='password'><br>

<input type='submit' value='OK'>

</form>

</div>

**4.login.php 登陆处理**

<?php

include\_once 'redis.php';

if(isset($\_POST['username'])){

$username=$\_POST['username'];

$password=$\_POST['password'];

//根据用户名取出用户ID

$id=$redis->get('user:'.$username);

if($id){

//根据ID在哈希中取出密码

if($password==$redis->hget('user:'.$id,'password')){

session\_start();

$\_SESSION['uid']=$id;

$\_SESSION['username']=$username;

header('location:index.php');

}else{

echo '密码错误！<br>';

}

}else{

echo '查无此人<br>';

}

}

?>

<div align='center'>

<h1>用户登录</h1>

<form action='login.php' method='post'>

名字：<input type='text' name='username'><br>

密码：<input type='password' name='password'><br>

<input type='submit' value='OK'>

</form>

</div>

**5.edit.php 修改处理**

<?php

include\_once 'redis.php';

if(isset($\_GET['id'])){

$id=$\_GET['id'];

//从哈希中获得id是$id的数据

$row=$redis->hgetall('user:'.$id);

//取出用户的名字

$edit\_name=$row['username'];

session\_start();

$\_SESSION['edit\_name']=$edit\_name;

}

if(isset($\_POST['id'])){

$id=$\_POST['id'];

$username=$\_POST['username'];

$password=$\_POST['password'];

//删除字符串中记录的用户

session\_start();

$redis->del('user:'.$\_SESSION['edit\_name']);

unset($\_SESSION['edit\_name']);

//增加字符串记录的用户

$redis->set('user:'.$username,$id);

//修改(覆盖)数据

$redis->hmset('user:'.$id,array('id'=>$id,'username'=>$username,'password'=>$password));

header('location:index.php');

}

?>

<div align='center'>

<h1>修改用户</h1>

<form action='edit.php' method='post'>

名字：<input type='text' name='username' value='<?echo $row['username']?>'><br>

密码：<input type='password' name='password' value='<?echo $row['password']?>'><br>

<input type='hidden' name='id' value='<?echo $row['id']?>'>

<input type='submit' value='OK'>

</form>

</div>

**6.del.php 删除处理**

<?php

include\_once 'redis.php';

$id=$\_GET['id'];

//查出要删除的用户名

$username=$redis->hget('user:'.$id,'username');

//删除字符串存储的用户名

$redis->del('user:'.$username);

//删除链表存储的ID

$redis->lrem('uid',$id);

//删除哈希中的数据

$redis->del('user:'.$id);

header('location:index.php');

?>

**7.delall.php 清空处理**

<?php

include\_once 'redis.php';

//清空数据库

$redis->flushdb();

session\_start();

session\_destroy();

header('location:index.php');

?>

**8.care.php 关注处理**

<?php

session\_start();

isset($\_SESSION['uid']) or exit();

include\_once 'redis.php';

$id=$\_GET['id'];

//集合中添加关注的用户ID

$redis->sadd('care\_user:'.$\_SESSION['uid'],$id);

header('location:index.php');

?>

# 十一、redis常用命令

1.string(字符串)

增/改

set key val

删

del key

查

get key

2.list(链表)

长度(php)

lsize('key')

增

rpush key1 val1

rpush key1 val2

rpush key1 val3 ......

删(php中)

$redis->lrem("id",$id);

改

lset key val new\_val

查

lrange key 0(头) -1(尾)

3.set(集合)

增

sadd key1 val1

sadd key1 val2

sadd key1 val3 ......

删

srem key val

改

查

smembers key

4.hash(哈希)

增

(单个下标)

hset key index1 val

hset key index2 val

hset key index3 val ......

(所有下标)

hmset key index1 val index2 val index3 val ......

删

del key

改

hset key index val

查

(单个下标)

hget key index

(所有下标)

hgetall key

系统

auth 密码

flushdb 清空数据

incr key 自增长