//验证是否安装OK:打开dos窗口->输入redis-server 在打开一个dos窗口-->输入redis-cli 显示：127.0.0.1:6379>则成功！  
 $redis = new redis();  
 $redis->connect('127.0.0.1', '6379') || die("连接失败！");  
 $redis->auth("admin"); //授权  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*string类型\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 //String 是最简单的类型，一个Key对应一个Value，String类型是二进制安全的。Redis的String可以包含任何数据，比如jpg图片或者序列化的对象   
 //$redis->set('test', 200); //赋值   
 //$redis->mset(array('key0' => 'value0', 'key1' => 'value1'));//批量赋值  
 //$redis->mget(array('key0', 'key1', 'key2'));//批量获取值   
 //$redis->setnx('key',value);//设置key对应的值为string类型的value,如果key已经存在，返回0,nx是not exist的意思  
 //$redis->msetnx(array('key0' => 'value0', 'key1' => 'value1'));//全部成功返回1，否则失败一个也返回0  
 //$redis->getset("key",'newvalue');//getset设置key的值，并返回key的旧值  
 //$redis->setex('key',10,'value10');//带生存时间的写入值单位s  
 //$redis->setrange('key',0,'hello');//设置子字符串替换，key，开始位置，替换值  
 //$redis->getrange("key",0,2);//getrange获取子字符串，getrange key start end   
 //$redis->ttl('key');//获取key的生存时间,没设置的为-1  
 //$redis->keys("\*a\*");//获取所有key，也可获取具体key，模糊匹配key  
 //$redis->get("test"); //根据key获取value  
 //$redis->getMultiple(array('key1', 'key2', 'key3'));  
 //$exits = $redis->exists('test'); //判断key是否存在。存在 true 不存在 false  
 //echo $exists?$res2:"null";  
 // echo $res1?$res2:"null";die;   
 //$redis->delete('key1', 'key2'); //删除,可删除多个  
 //$redis->delete(array('key3', 'key4', 'key5'));//删除,可删除多个  
 //$redis->delete( $redis->keys("\*"));//传数组，删除所有key  
 //$redis->incr("key",1);//自增，默认值1   
 //$redis->incrBy('key',10);//必须给定参数值  
 //$redis->decr("key",10);//自减，默认值1   
 //$redis->decrBy('key',10);//必须给定参数值  
 //$redis->append("key","appendvalue");//给指定的key的字符串追加value,返回新字符串的长度  
 //$redis->strlen('key');//获取指定key的字符串长度  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*string类型\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*hash类型\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 // Redis hash是一个string类型的field和value的映射表。它的添加删除都是0(1)  
 // hash特别适合用于存储对象。相对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。  
 // $redis->flushall();die;//清空所有数据库  
 // $redis->flushdb();die;//清空当前数据库  
   
   
 //$redis->hset("test","username","admin");//单个设置  
 //$redis->hget("test","username");//单个获取  
 //$redis->hsetnx("test","username","admin@qq.com");//设置hash field 为指定的值value,如果feild已经存在，返回0,nx是not exist的意思   
 //$redis->hmset("testabc",array("a"=>1,"b"=>2,"c"=>3,'d'=>4));//批量设置  
 //$redis->hmget("testabc",array("a","b","c"));//批量获取  
 //$redis->hincrBy("test","password","123456");//设置hash field 添加key和value  
 //$redis->hexists("test","usernames");//测试hash field 存在1，不存在0  
 //$redis->hlen("test"); //返回hash 里所有的 field 的数量  
 //$redis->hdel("test","key");//删除指定hash的field  
 //$redis->hkeys("test"); //返回hash 所有 field  
 //$redis->hvals("test");//返回hash 所有 field values  
 //$redis->hgetall("test");//获取hash中全部的field和value  
   
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*hash类型\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*list链表\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 //輚先进后出；队列先进先出；  
 //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既可以作为栈，又可以作为队列。  
 //$redis->lrange("list",0,-1);//取链表list 从0位开始取到最后位置；0代表头 -1代表尾  
 //$redis->lpush("list","1"); //在key对应的list的头部添加字符串元素  
 //$redis->rpush("list","test"); //在key对应的list的尾部添加字符串元素  
 //$redis->lpop("list");//在key对应的list的头部删除字符串元素，并返回删除字符  
 //$redis->rpop("list");//在key对应的list的尾部删除字符串元素，并返回删除字符  
 // echo $redis->lrem("list",2,'key');//在list中删除N个和value相同的元素(n<0从尾删除,n=0全部删除),返回的值为删除的个数  
 //$redis->lset("list",-1,"啊");//设置list中指定下标的元素值 lset(list,位置，值)，位置可以从头或者从尾  
 //$redis->lindex("list",0);//返回list中index位置的元素  
 //$redis->llen("list");//返回list的长度  
 //$redis->ltrim("list",0,1);//保留指定key的值的范围内的数据 0为头 -1为尾   
 //$redis->linsert("list");//\*\*\*\*\*\*\*\*  
 //$redis->rpoplpush("list","listt");//从第一个LIST的尾部移除元素并添加到第二个LIST的头部  
   
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*list链表\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*set集合\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 // 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，删除和查找的复杂程度都是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  
 //$redis->sadd("col","a","b","c");//向集合中添加元素  
 //$redis->smembers("col");//集合元素列表  
 //$redis->srem("col",'d');//集合移除元素  
 //$redis->spop("col");//随机删除集合元素并返回删除的元素。删除最后一个元素时，则集合也被删除  
 //$redis->sdiff("set1","set2");//取多个集合的差集，谁在前面已谁为标准（差集就是前面第一个集合有而后面所有集合都没有）  
 //$redis->sdiffstore("dif","set1","set2");//返回多个集合的差集，存在第三个集合里面  
 //$redis->sinter("set","set1",'set2');//交集（所有集合都有的元素）  
 //$res = $redis->sinterstore("inter","set","set1");///取多个集合的交集,存在第三个集合里面  
 //$redis->sunion("set","set1",'set2');//并集（所有集合合并去重的元素）  
 //$redis->sunionstore("union","set","set1");///取多个集合的并集,存在第三个集合里面  
 //$redis->smove("set1","set2","value");///第一个集合元素剪切到第二个集合里面。  
 //$redis->scard("set");//返回集合中元素的个数  
 //$redis->sismember("set","value");//判断某个元素是否为集合的元素  
 //$redis->srandmember("set");//随机返回集合内的一个元素，不会删除元素  
   
// $redis->keys("\*"); //返回所有key  
// $redis->keys("\*t"); //返回所有s开头的key  
// $res = $redis->keys("\*");  
// foreach ($res as $k => $v) {  
// echo $v."=>".$redis->get($v)."<br>";  
// }  
// $redis->exists("set"); //判断该key是否存在  
// $redis->del("inter"); //删除该key   
// $redis->expire("set2",100); //设置key 过期时间10s  
// $redis->ttl("set3");//查看过期剩余时间s  
   
 //$redis->select("1");// 选择数据库  
 //$redis->set("name","test");// 设置key=>value  
 //$redis->get("name");// 根据key获取value  
 //$redis->move("get",1);// 把key移到另一个数据库  
 //$redis->persist("set2");// 移除过期时间  
 //$redis->randomkey();//随机返回一个key  
 //$redis->rename("names","newname");//重命名key  
 //$redis->type("zset");//获取key类型  
 //$redis->dbsize();//获取数据库key个数  
 //$redis->info();//获取redis服务信息  
 //$redis->echo("hello，world");//输出函数  
   
   
   
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*set集合\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 //redis有五大数据类型  
 //字符串 string  
 //hash 表  
 //链表结构 list  
 //set 集合(无序)  
 //zset 有序集合  
   
 //string来存储用户的最大id,用来id的自增   
 //hash 表来存储用户的详细信息   
 //list链表结构来存储用户的id号  
   
// $uid = $redis->incr("blog:id"); //自增长  
// $res = $redis->hmset('blog:user:' . $uid, array('id' => $uid, 'username' => 'admin'.time() . time(), 'password' => md5(time()))); //hash表插入一条记录  
// $redis->rpush("blog:uid", $uid);  
//   
//   
// $res = $redis->hgetall("blog:user:".$uid);  
// $redis->del("blog:user:".$uid);//删除用户  
// $redis->lrem("blog:uid",$uid);//删除链表中的id  
 // print\_R($res);die;  
   
  
// $count = $redis->lsize("blog:uid"); //list类型count总数   
// $page\_size = 3;  
// $page\_num = !empty($\_GET['page']) ? $\_GET['page'] : 1;  
// $page\_count = ceil($count / $page\_size);  
// $ids = $redis->lrange("blog:uid", ($page\_num - 1) \* $page\_size, (($page\_num - 1) \* $page\_size + $page\_size - 1));  
//  
// foreach ($ids as $v) {  
//  
// $ress[] = $redis->hgetall("blog:user:" . $v); //获取记录   
// }  
// echo"<pre>";  
// print\_R($ress);

作者：DragonersLi

链接：http://www.jianshu.com/p/c1c798828477

來源：简书

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。