

redis课件

Redis 是一个开源的使用 ANSI C 语言编写、遵守 BSD 协议、支持网络、可基于内存、分布式、可选持久性的键值对(Key-Value)存储数据库，并提供多种语言的 API。Redis 通常被称为数据结构服务器，因为值 (value) 可以是字符串(String)、哈希(Hash)、列表(list)、集合(sets)和有序集合(sorted sets)等类型。

Redis 与其他 key - value 缓存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类型的数据，同时还提供list, set, zset, hash等数据结构的存储。
- Redis支持数据的备份，即master-slave模式的数据备份。

redis优势

- 性能极高 – Redis能读的速度是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 – Redis的所有操作都是原子性的，意思就是要么成功执行要么失败完全不执行。单个操作是原子性的。多个操作也支持事务，即原子性，通过MULTI和EXEC指令包起来。
- 丰富的特性 – Redis还支持 publish/subscribe, 通知, key 过期等等特性。

Redis与其他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，这是一个不同于其他数据库的进化路径。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.redis安装

- windows下

1. 下载地址 <https://github.com/tporadowski/redis/releases>

▼ Assets 4

 Redis-x64-5.0.10.msi	7.89 MB
 Redis-x64-5.0.10.zip	14.4 MB
 Source code (zip)	
 Source code (tar.gz)	

2. 解压后的文件目录如下：


```
C:\Users\lixio>redis-cli
127.0.0.1:6379>
```

- mac下

1. 下载地址 <https://download.redis.io/releases/redis-6.2.4.tar.gz>, 如果不好下载就在群文件下载吧

2. 安装

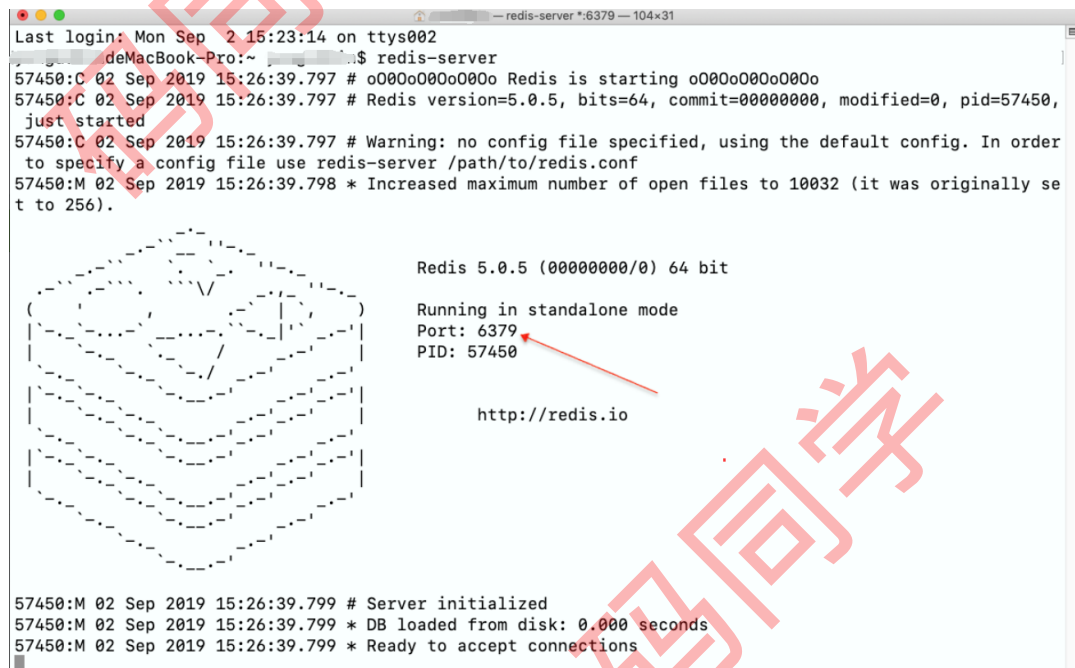
在终端进入下载后的目录, 然后:

- 解压: tar zxvf redis-6.2.4.tar.gz
- 移动到: sudo mv redis-6.2.4 /usr/local
- 切换到: cd /usr/local/redis-6.2.4/
- 编译测试: make test
- 编译安装: make install

完成安装

3. 开启redis服务端

要使用redis, 先开启redis服务端, 在终端输入redis-server, 如下:



```
deMacBook-Pro:~$ redis-server
57450:C 02 Sep 2019 15:26:39.797 # o000o000o000o Redis is starting o000o000o000o
57450:C 02 Sep 2019 15:26:39.797 # Redis version=5.0.5, bits=64, commit=00000000, modified=0, pid=57450,
just started
57450:C 02 Sep 2019 15:26:39.797 # Warning: no config file specified, using the default config. In order
to specify a config file use redis-server /path/to/redis.conf
57450:M 02 Sep 2019 15:26:39.798 * Increased maximum number of open files to 10032 (it was originally se
t to 256).
```

Redis 5.0.5 (00000000/0) 64 bit

Running in standalone mode

Port: 6379

PID: 57450


<http://redis.io>

```
57450:M 02 Sep 2019 15:26:39.799 # Server initialized
57450:M 02 Sep 2019 15:26:39.799 * DB loaded from disk: 0.000 seconds
57450:M 02 Sep 2019 15:26:39.799 * Ready to accept connections
```

可以看到redis服务端默认在6379端口成功开启, 不要关闭此窗口

4. 连接redis

重新打开一个终端, 输入redis-cli, 打开redis客户端



```
deMacBook-Pro:~$ redis-cli
127.0.0.1:6379>
```

2.redis基础

Redis支持五种数据类型：string（字符串），hash（哈希），list（列表），set（集合）及zset(sorted set：有序集合）。

首先启动redis服务，并启动redis客户端

- 字符串

```
127.0.0.1:6379> set key1 value1 #设置一个键值对数据，key是key1，value是value1
OK
127.0.0.1:6379> get key1 #获取key1对应的value
"value1"
```

- 哈希

Redis hash 是一个 string 类型的 field（字段）和 value（值）的映射表，hash 特别适合用于存储对象。

```
# hmset是设置哈希值的命令，设置一个key为hkey的哈希键值对，哈希里包括里三组值
name=shamo,age=18,job=tester
127.0.0.1:6379> hmset hkey name shamo age 18 job tester
OK
127.0.0.1:6379> hgetall hkey #hgetall是获取某个哈希key对应的所有值
1) "name"
2) "shamo"
3) "age"
4) "18"
5) "job"
6) "tester"
127.0.0.1:6379> hget hkey name #hget是获取某个哈希key对应的某组key对应的值
"shamo"
```

如果以表格形式展示的话，那么hkey对应的数据就相当于下图

row	key	value
1	name	shamo
2	age	18
3	job	tester

- 列表

Redis列表是简单的字符串列表，按照插入顺序排序。你可以添加一个元素到列表的头部（左边）或者尾部（右边）

一个列表最多可以包含 $2^{32}-1$ 个元素 (4294967295, 每个列表超过40亿个元素)。

```
# lpush是插入列表值的命令，向key为teachers的列表中插入shamo,beihe,lou
127.0.0.1:6379> lpush teachers shamo beihe lou
(integer) 3
# lrange是获取列表指定范围内的元素
127.0.0.1:6379> lrange teachers 0 3
1) "lou"
2) "beihe"
3) "shamo"
```

如果以表格形式展示的话，那么teachers对应的数据就相当于下图

row	value
1	lou
2	beihe
3	shamo

- 集合

Redis 的 Set 是 String 类型的无序集合。集合成员是唯一的，这意味着集合中不能出现重复的数据。

集合对象的编码可以是 intset 或者 hashtable。

```
# sadd 命令向名为 languages 的集合插入的三个元素
127.0.0.1:6379> sadd languages java python javascript
(integer) 3
127.0.0.1:6379> smembers languages # smembers 返回集合中的所有成员
1) "javascript"
2) "java"
3) "python"
```

- 有序集合

Redis 有序集合和集合一样也是 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。

有序集合的成员是唯一的,但分数(score)却可以重复。

```
#zadd 向 redis 的有序集合中添加了四个值并关联上分数
127.0.0.1:6379> zadd students 4 xueyuan4 3 xueyuan3 1 xueyuan1 2 xueyuan2
(integer) 4
127.0.0.1:6379> zrange students 0 10 # zrange 通过索引区间返回有序集合指定区间内的成员
1) "xueyuan1"
2) "xueyuan2"
3) "xueyuan3"
4) "xueyuan4"
```

如果以表格形式展示的话，那么teachers对应的数据就相当于下图

row	value	score
1	xueyuan1	1
2	xueyuan2	2
3	xueyuan3	3
4	xueyuan4	4

- 有效时间相关命令

```
127.0.0.1:6379> expire students 60 #expire 设置某个key的超时时间，单位是秒
(integer) 1
127.0.0.1:6379> ttl students # ttl #命令是以秒为单位，查看某个key有效时间还有多久
(integer) 57
127.0.0.1:6379> pexpire name 1000000 #pexpire 设置某个key的超时时间，单位是毫秒
(integer) 1
127.0.0.1:6379> pttl name # pttl #命令是以毫秒为单位，查看某个key有效时间还有多久
(integer) 996834
```

3.python操作redis

1. 第三方模块redis按照

```
pip install redis
```

2. 连接redis

```
import redis

# host是redis主机，需要redis服务端和客户端都起着 redis默认端口是6379
pool = redis.ConnectionPool(host='localhost', port=6379,
decode_responses=True)
r = redis.Redis(connection_pool=pool)
```

3. 操作字符串

```
# key是"gender" value是"male" 将键值对存入redis缓存,有效期300秒
r.set('gender', 'male',ex=300)
# key是"name" value是"shamo" 将键值对存入redis缓存,有效期30000毫秒
r.set('name', 'shamo',px=30000)
# 取出键gender对应的值
print(r.get('gender'))
#批量获取，取出键gender、name对应的值
print(r.mget('gender', 'name'))
```

4. 操作哈希

```
# 设置一个哈希"user1",其中包括name=shamo,age=18,job=tester
r.hset('user1','name','shamo')
r.hset('user1','age',18)
r.hset('user1','job','tester')
# 获取user1对应的所有值
print(r.hgetall('user1'))
# 批量设置哈希"user2",其中包括name=shamo,age=18,job=tester
r.hmset('user2',{'name':'shamo','age':18,'job':'tester'})
# 批量获取哈希"user2"中的key为name和age的值
print(r.hmget('user2','name','age'))
```

5. 操作列表

```
# 向列表"list1"中追加数据data1, data2
r.lpush('list1','data1','data2')
# 获取列表"list1"中所有数据
print(r.lrange('list1',0,-1))
```

6. 操作集合(set)

```
# 向列表"set1"中追加数据data1, data2
r.sadd('set1','data1','data2')
# 获取集合"set1"的所有元素
print(r.smembers('set1'))
```

7. 操作有序集合

```
# 向有序集合"zset1"中追加数据data1,data2, 其中数字是各自数据的分数, 用来排序
r.zadd('zset1',{'data1':14,'data2':13})
# 获取"zset1"中所有数据
print(r.zrange('zset1',0,-1))
```

8. 其他操作

```
r.delete('key1') #删除某个key
r.exists('key1') #检查key是否存在
r.expire('user1',10) #设置这个key的有效期为10秒
r.pexpire('user2',10000) #设置这个key的有效期为10000毫秒
r.type('zset1') #获取某个key对应的数据类型
```

4.商城项目redis实战

在商城项目的接口中,存在着一些接口是将数据存储在redis缓存里,因此接口测试中要做数据校验的话就必须实施对redis的数据获取操作。

商城项目后台采用的是java代码实现,所以在redis缓存中存储的基本都是java对象的序列化数据,那么python在获取java对象的序列化数据后需要进行反序列化转成对象才能正常获取数据信息,因此安装一个第三方库,用来说java对象的转换

1. 安装第三方库javaobj

```
pip install javaobj-py3
```

2. 封装redis

```

class RedisUtil:
    def __init__(self, host, pwd, port=6379, decode_responses=False):
        """
        :param host: redis服务器地址
        :param pwd: redis密码
        :param port: redis端口 默认6379
        :param decode_responses: 默认为False, 表示返回的是bytes数据
        """
        self.pool = redis.ConnectionPool(host=host,
        port=port, password=pwd, encoding_errors='ignore', decode_responses=decode_respon
        ses)
        self.r = redis.Redis(connection_pool=self.pool)
    def get(self, key):
        type = self.r.type(key).decode('utf8')
        if type=='string':
            return self.r.get(key)
        elif type=='hash':
            return self.r.hgetall(key)
        elif type=='zset':
            return self.r.zrange(key, 0, -1)
        elif type=='set':
            return self.r.smembers(key)
        else:
            raise Exception(f'不支持的数据类型{type}或者{key}不存在')

```

3. 实战立即购买接口

立即购买接口在请求成功后, 响应数据为空, 并且会将购买产品的数据存入redis

key值为{BUY_NOW_ORIGIN_DATA_PREFIX}_59, 其中的59是用户id, 每个用户存储的key会根据用户id而变化

存储的数据结构是java中的ArrayList对象的序列化字符串, 并且集合中存储的是立即购买的产品对象信息

```

rd = RedisUtil(host='121.42.15.146', pwd='testfan')
res = rd.get('{BUY_NOW_ORIGIN_DATA_PREFIX}_59')
# 将从redis中得到的序列化java对象转换成python对象
res1 = javaobj.loads(res) #res1是个列表
# 得到列表里第一个购买对象的数量
print(res1[0].__getattribute__('num'))
# 得到列表里第一个购买对象的产品id
print(res1[0].__getattribute__('skuId'))

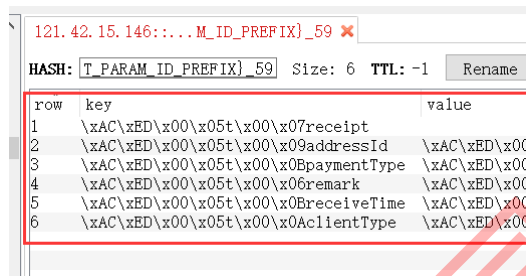
```

4. 实战设置收货地址接口

设置收货地址接口接口在请求成功后, 响应数据为空, 并且会将地址id数据存入redis

key值为{CHECKOUT_PARAM_ID_PREFIX}_59, 其中的59是用户id, 每个用户存储的key会根据用户id而变化

这个key在redis中对应的值是个hash值, 分别存储的是立即购买后的收货地址、付款方式、配送清单、送货时间数据



row	key	value
1	\xAC\xED\x00\x05t\x00\x07receipt	
2	\xAC\xED\x00\x05t\x00\x09addressId	\xAC\xED\x00
3	\xAC\xED\x00\x05t\x00\x0BpaymentType	\xAC\xED\x00
4	\xAC\xED\x00\x05t\x00\x06remark	\xAC\xED\x00
5	\xAC\xED\x00\x05t\x00\x0BreceiveTime	\xAC\xED\x00
6	\xAC\xED\x00\x05t\x00\x0AclientType	\xAC\xED\x00

并且在hash值里每个key对应的java数据对象可能还不一样，可以采用如下代码手段处理

```
rd = RedisUtil(host='121.42.15.146',pwd='testfan')
res = rd.get('{CHECKOUT_PARAM_ID_PREFIX}_59')
# print(type(res))
# res是个hash值，对应着python里的字典，所以按照字典进行遍历
for key,value in res.items():
    # 将key转换
    key1 = javaobj.loads(key)
    print('key:{}'.format(key1))
    #由于key对应的类型并不确定，所以在这里做了异常处理
    try:
        # 将value转换
        value1 = javaobj.loads(value)
        print('value1:{}'.format(value1))
    except:
        pass
```

5. 接口测试中redis数据断言

将第3/4步的代码进行封装处理返回redis中查到的数据，和期望数据做对比即可完成