理论上来讲，主要key或者value能被序列化，都可以存储在memcache中

Memcache的两种用途：

1. 做缓存
2. 应用之间交互的数据存储

Memcache参考：<http://blog.csdn.net/jinxingfeng_cn/article/details/24264693>

**1、启动Memcache 常用参数**  
memcached 1.4.3  
-p <num>      设置端口号(默认不设置为: 11211)  
-U <num>      UDP监听端口 (默认: 11211, 0 时关闭)    
-l <ip\_addr>  绑定地址 (默认:所有都允许,无论内外网或者本机更换IP，有安全隐患，若设置为127.0.0.1就只能本机访问)  
-d            duli进程运行  
-u <username> 绑定使用指定用于运行进程 <username>  
-m <num>      允许最大内存用量，单位M (默认: 64 MB)  
-P <file>     将PID写入文件<file>，这样可以使得后边进行快速进程终止, 需要与 -d 一起使用  
如：  
在**[Linux](http://lib.csdn.net/base/linux" \o "Linux知识库" \t "http://blog.csdn.net/jinxingfeng_cn/article/details/_blank)**下：./usr/local/bin/memcached -d -u jb-mc -l 192.168.1.197 -m 2048 -p 12121  
在window下：d:\App\_Serv\memcached\memcached.exe -d RunService -l 127.0.0.1 -p 11211 -m 500  
在windows下注册为服务后运行：  
sc.exe create jb-Memcached binpath= “d:\App\_Serv\memcached\memcached.exe -d RunService -p 11211 -m 500″ start= auto  
net start jb-Memcached

**2、telnet连接**

telnet 127.0.0.1 11211

**3、写入memcache**

**3.1 memcached Telnet Interface**

|  |  |  |
| --- | --- | --- |
| **Command** | **Description** | **Example** |
| get | Reads a value | get mykey |
| set | Set a key unconditionally | set mykey 0 60 5 |
| add | Add a new key | add newkey 0 60 5 |
| replace | Overwrite existing key | replace key 0 60 5 |
| append | Append data to existing key | append key 0 60 15 |
| prepend | Prepend data to existing key | prepend key 0 60 15 |
| incr | Increments numerical key value by given number | incr mykey 2 |
| decr | Decrements numerical key value by given number | decr mykey 5 |
| delete | Deletes an existing key | delete mykey |
| flush\_all | Invalidate specific items immediately | flush\_all |
| Invalidate all items in n seconds | flush\_all 900 |
| stats | Prints general statistics | stats |
| Prints memory statistics | stats slabs |
| Prints memory statistics | stats malloc |
| Print higher level allocation statistics | stats items |
|  | stats detail |
|  | stats sizes |
| Resets statistics | stats reset |
| version | Prints server version. | version |
| verbosity | Increases log level | verbosity |
| quit | Terminate telnet session | quit |

查看版本

version

VERSION 1.4.4-14-g9c660c0

set num 0 0 2

12

STORED

incr num

13

decr num 1

12

flush\_all

OK

**3.2 telnet请求命令格式**

<command name> <key> <flags> <exptime> <bytes>\r\n <data block>\r\n  
a) <command name> 可以是”set”, “add”, “replace”。  
“set”表示按照相应的<key>存储该数据，没有的时候增加，有的覆盖。  
“add”表示按照相应的<key>添加该数据,但是如果该<key>已经存在则会操作失败。  
“replace”表示按照相应的<key>替换数据,但是如果该<key>不存在则操作失败

b) <key> 客户端需要保存数据的key。

c) <flags> 是一个16位的无符号的整数(以十进制的方式表示)。  
该标志将和需要存储的数据一起存储,并在客户端get数据时返回。  
客户可以将此标志用做特殊用途，此标志对服务器来说是不透明的。

d) <exptime> 过期的时间。  
若为0表示存储的数据永远不过时(但可被服务器**[算法](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "http://blog.csdn.net/jinxingfeng_cn/article/details/_blank)**：LRU 等替换)。  
如果非0(unix时间或者距离此时的秒数),当过期后,服务器可以保证用户得不到该数据(以服务器时间为标准)。

e) <bytes> 需要存储的字节数(不包含最后的”\r\n”),当用户希望存储空数据时,<bytes>可以为0

f) 最后客户端需要加上”\r\n”作为”命令头”的结束标志。  
<data block>\r\n

紧接着”命令头”结束之后就要发送数据块(即希望存储的数据内容),最后加上”\r\n”作为此次通讯的结束。

**3.3 telnet响应命令**

结果响应：reply  
当以上数据发送结束之后,服务器将返回一个应答。可能有如下的情况:

a) “STORED\r\n”：表示存储成功  
b) “NOT\_STORED\r\n” ： 表示存储失败,但是该失败不是由于错误。  
通常这是由于”add”或者”replace”命令本身的要求所引起的,或者该项在删除队列之中。

如： set key 33 0 4\r\n  
ffff\r\n

**4、获取/检查KeyValue**  
get <key>\*\r\n  
a) <key>\* 表示一个或者多个key(以空格分开)  
b) “\r\n” 命令头的结束

结果响应：reply  
服务器端将返回0个或者多个的数据项。每个数据项都是由一个文本行和一个数据块组成。当所有的数据项都接收完毕将收到”END\r\n”  
每一项的**[数据结构](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "http://blog.csdn.net/jinxingfeng_cn/article/details/_blank)**：  
VALUE <key> <flags> <bytes>\r\n  
<data block>\r\n

a) <key> 希望得到存储数据的key  
b) <falg> 发送set命令时设置的标志项  
c) <bytes> 发送数据块的长度(不包含”\r\n”)  
d) “\r\n” 文本行的结束标志  
e) <data block> 希望接收的数据项。  
f) “\r\n” 接收一个数据项的结束标志。

如果有些key出现在get命令行中但是没有返回相应的数据，这意味着服务器中不存在这些项，这些项过时了，或者被删除了  
如：get aa  
VALUE aa 33 4  
ffff  
END

**5、删除KeyValue：**  
delete <key> <time>\r\n

a) <key> 需要被删除数据的key  
b) <time> 客户端希望服务器将该数据删除的时间(unix时间或者从现在开始的秒数)  
c) “\r\n” 命令头的结束

delete foo

DELETED

**6、检查Memcache服务器状态：**  
stats\r\n  
在这里可以看到memcache的获取次数，当前连接数，写入次数，已经命中率等；

pid ： 进程id  
uptime ：总的运行时间，秒数  
time ： 当前时间  
version ： 版本号  
……  
curr\_items ： 当前缓存中的KeyValue数量  
total\_items ： 曾经总共经过缓存的KeyValue数量  
bytes ： 所有的缓存使用的内存量  
curr\_connections 当前连接数  
….  
cmd\_get ： 总获取次数  
cmd\_set ： 总的写入次数  
get\_hits ： 总的命中次数  
miss\_hits :  获取失败次数  
…..  
bytes\_read ： 总共读取的流量字节数  
bytes\_written ： 总的写入流量字节  
limit\_maxbytes ： 最大允许使用的内存量，字节

**7、高级缓存细节查看方法：**  
stats reset  
清空统计数据

stats malloc  
显示内存分配数据

stats cachedump slab\_id limit\_num  
显示某个slab中的前limit\_num个key列表，显示格式如下  
ITEM key\_name [ value\_length b; expire\_time|access\_time s]  
其中，memcached 1.2.2及以前版本显示的是  访问时间(timestamp)  
1.2.4以上版本，包括1.2.4显示 过期时间(timestamp)  
如果是永不过期的key，expire\_time会显示为服务器启动的时间

stats cachedump 7 2  
ITEM copy\_test1 [250 b; 1207795754 s]  
ITEM copy\_test [248 b; 1207793649 s]

stats slabs  
显示各个slab的信息，包括chunk的大小、数目、使用情况等

stats items  
显示各个slab中item的数目和最老item的年龄(最后一次访问距离现在的秒数)

stats items

STAT items:1:number 2

STAT items:1:age 3054542729

STAT items:1:evicted 0

STAT items:1:evicted\_nonzero 0

STAT items:1:evicted\_time 0

STAT items:1:outofmemory 0

STAT items:1:tailrepairs 0

END

stats detail [on|off|dump]  
设置或者显示详细操作记录

参数为on，打开详细操作记录  
参数为off，关闭详细操作记录  
参数为dump，显示详细操作记录(每一个键值get、set、hit、del的次数)

**8、清空所有键值**  
flush\_all  
注：flush并不会将items删除，只是将所有的items标记为expired，因此这时memcache依旧占用所有内存。

**9、退出**  
quit\r\n