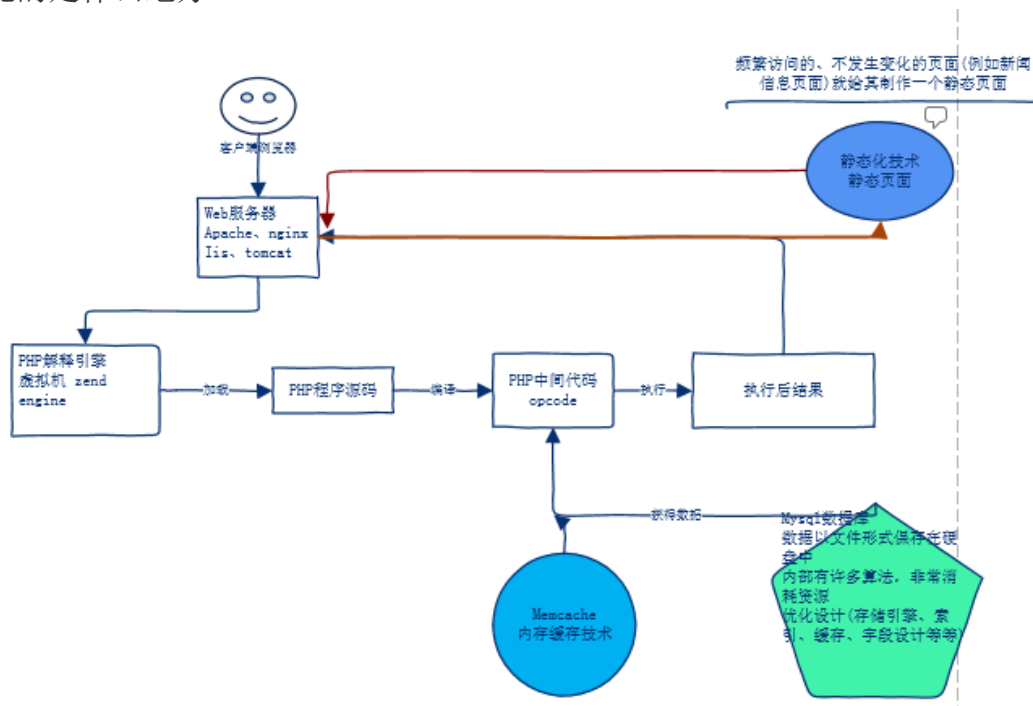


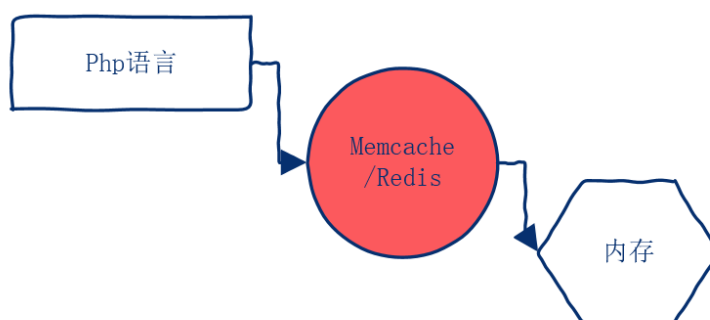
1 系统核心优化

思路：以小博大、利用最小的资源换取最大的回报
memcache、mysql 优化、静态化技术
优化的是什么地方？



2 Memcached

memory cache 内存缓存技术



Memcache 或 Redis 是中介介质，可以帮助我们通过 php 语言实现对内存的操作

2.1 memcache 和 redis 的区别、联系

2.1.1 区别

前者：

每个 key 的数据最大是 1M

对各种技术支持比较全面，session 可以存储 memcache 中，各种框架(例如 thinkphp)对 memcache 支持的比较好

比较老牌、传统的内存缓存技术
适合存储简单、实用的数据
数据类型只有 String
没有持久化

后者:

每个 key 的数据最大是 1G
对各种技术支持没有 memcache 更好。
新兴的内存缓存技术
适合做集合计算(list/set/sortset)
数据类型较丰富 (String/list/Set/Sort set/hash)
有持久化

2.1.2 联系

数据存储在内存当中, 数据模型都是 key-value
两种内存缓存技术都要掌握, 大家有各自擅长的地方

memcache: 对 session 支持, 各种框架支持

redis: 集合计算

2.2 Windows 安装使用 memcached

2.2.1 安装 memcached

解压:

20151006-Memcache-01-核心内存概述.wmv	2015/10/6 9:55	媒体文件(.wmv)	21,014 KB	00:11:3
20151006-Memcache-02-与Redis的比较.wmv	2015/10/6 10:19	媒体文件(.wmv)	17,825 KB	00:11:3
Memcache-20151006.doc	2015/10/6 10:20	Microsoft Word ...	80 KB	
Memcached 原理和使用详解.pdf	2015/5/2 17:12	PDF Document	611 KB	
memcached.pdf	2015/5/2 17:12	PDF Document	930 KB	
memcached-1.2.6-win32-bin.zip	2015/5/2 17:12	360压缩 ZIP 文件	37 KB	
php_memcache.dll	2010/10/3 23:42	应用程序扩展	81 KB	

解压后生成文件(服务器端执行文件):

20151006-Memcache-02-与Redis的比较.wmv	2015/10/6 10:19	媒体文件(.wmv)	17,825 KB	00:11:3
Memcache-20151006.doc	2015/10/6 10:21	Microsoft Word ...	102 KB	
Memcached 原理和使用详解.pdf	2015/5/2 17:12	PDF Document	611 KB	
memcached.exe	2008/9/24 9:14	应用程序	84 KB	
memcached.pdf	2015/5/2 17:12	PDF Document	930 KB	
memcached-1.2.6-win32-bin.zip	2015/5/2 17:12	360压缩 ZIP 文件	37 KB	
php_memcache.dll	2010/10/3 23:42	应用程序扩展	81 KB	

考虑(服务器端)执行文件到 H: 目录:

Program Files	2015/10/6 10:55	文件夹		
Program Files (x86)	2015/10/3 8:45	文件夹		
memcached.exe	2008/9/24 9:14	应用程序	84 KB	
vmwaremachine	2015/7/24 9:08	文件夹		

2.2.2 开启 memcached 服务

前台开启服务(不推荐):

```
C:\Users\jinnan>H:
H:\>memcached
```

Ctrl+C 结束掉

开启服务可以设置的参数:

```
H:\>memcached -help
memcached 1.2.6
-p <num>      TCP port number to listen on <default: 11211>
-U <num>      UDP port number to listen on <default: 0, off>
-s <file>     unix socket path to listen on <disables network support>
-a <mask>     access mask for unix socket, in octal <default 0700>
-l <ip_addr>  interface to listen on, default is INADDR_ANY
-d start      tell memcached to start
-d restart    tell running memcached to do a graceful restart
-d stop!shutdown tell running memcached to shutdown
-d install    install memcached service
-d uninstall  uninstall memcached service
-r            maximize core file limit
-u <username> assume identity of <username> <only when run as root>
-m <num>     max memory to use for items in megabytes, default is 64 MB
-M           return error on memory exhausted <rather than removing items>
-c <num>     max simultaneous connections, default is 1024
-k           lock down all paged memory. Note that there is a
             limit on how much memory you may lock. Trying to
             allocate more than that would fail, so be sure you
             set the limit correctly for the user you started
             the daemon with <not for -u <username> user;
             under sh this is done with 'ulimit -S -l NUM_KB'.
-v           verbose <print errors/warnings while in event loop>
```

监听端口, 默认11211

监听ip地址

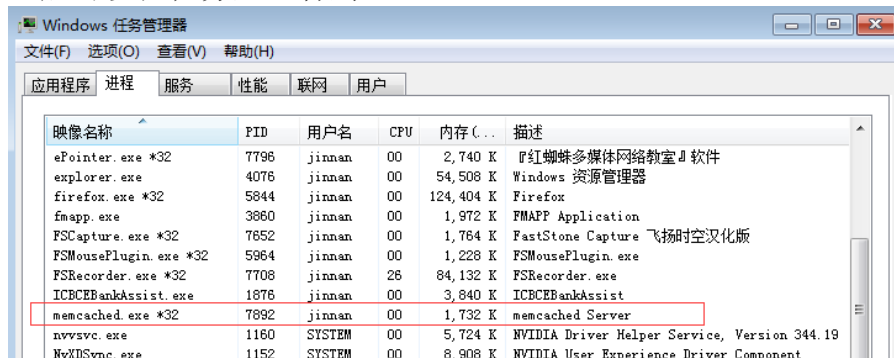
最大使用内存

通过设置参数开启 memcache 服务:

```
H:\>memcached -p 11213 -l 127.0.0.1 -m 128MB
```

端口11213、监听本机ip、一共限制使用128MB内存方式
开启memcache服务

之后可以在任务里边看到 memcache:



2.2.3 设置开机启动项服务

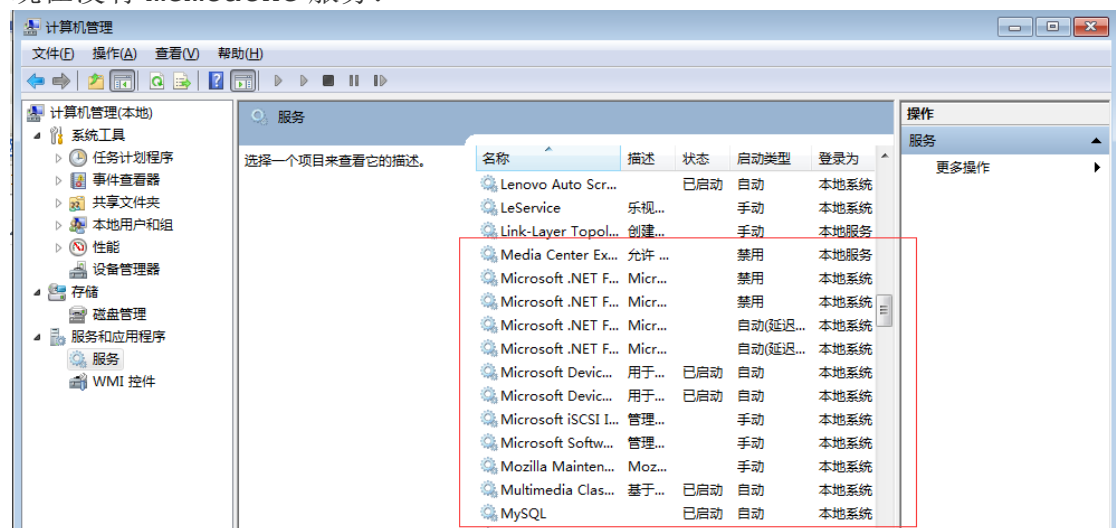
sc create 名称 svn 相关参数

```
H:\>memcached -help
memcached 1.2.6
-p <num>      TCP port number to listen on (default: 11211)
-U <num>      UDP port number to listen on (default: 0, off)
-s <file>     unix socket path to listen on (disables network support)
-a <mask>     access mask for unix socket, in octal (default 0700)
-l <ip_addr>  interface to listen on, default is INADDR_ANY
-d start      tell memcached to start
-d restart    tell running memcached to do a graceful restart
-d stop!shutdown tell running memcached to shutdown
-d install    install memcached service
-d uninstall  uninstall memcached service
-r            maximize core file limit
-u <username> assume identity of <username> (only when run as root)
-m <num>      max memory to use for items in megabytes, default is 64 MB
-M           return error on memory exhausted (rather than removing items)
-c <num>      max simultaneous connections, default is 1024
-k           lock down all paged memory. Note that there is a
            limit on how much memory you may lock. Trying to
            allocate more than that would fail, so be sure you
            set the limit correctly for the user you started
```

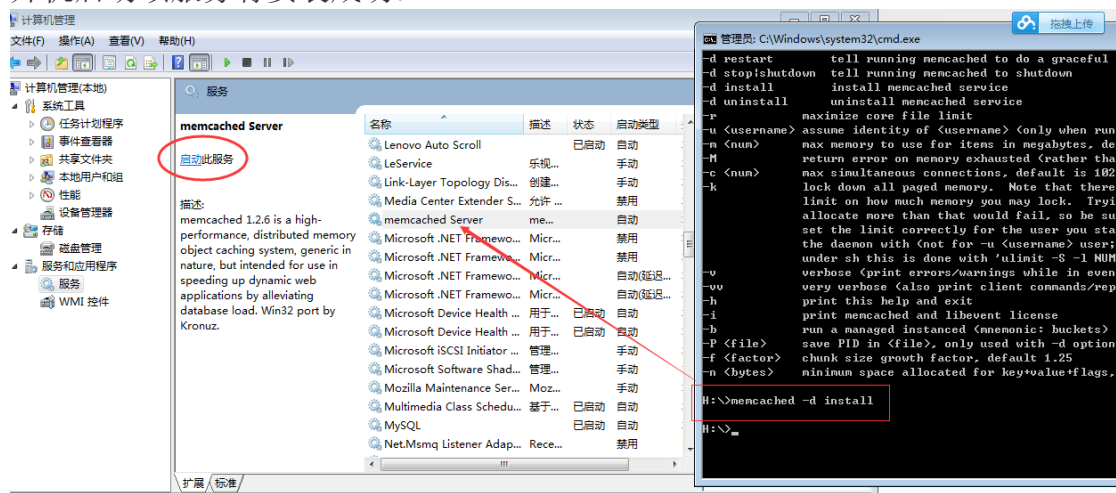
启动、关闭、重
启服务

安装、卸载服务

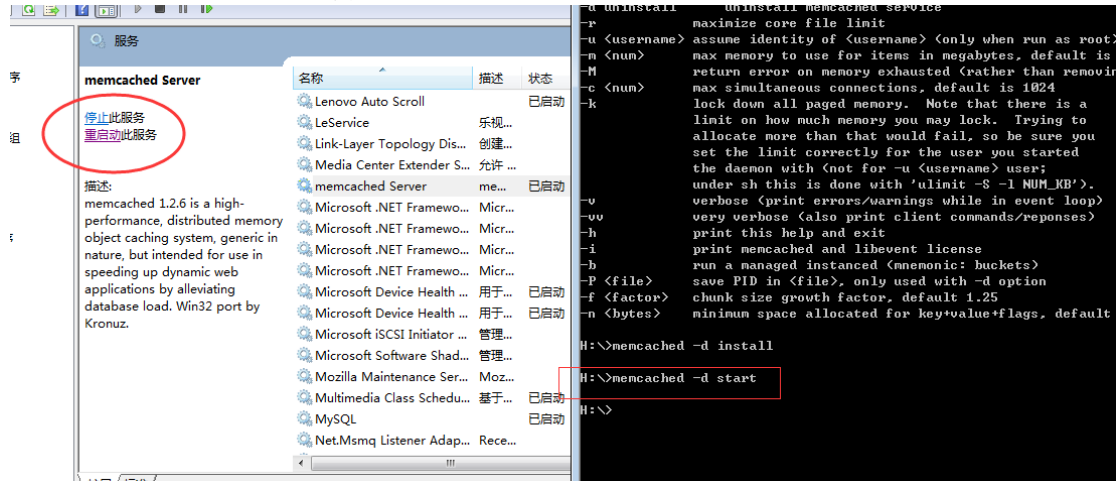
现在没有 memcache 服务:



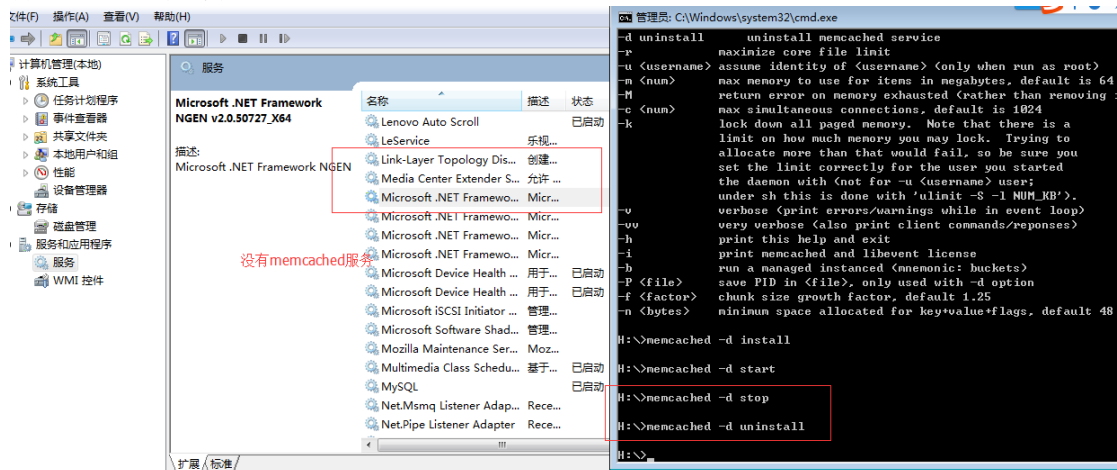
开机启动项服务有安装成功:



通过-d start 开启服务:

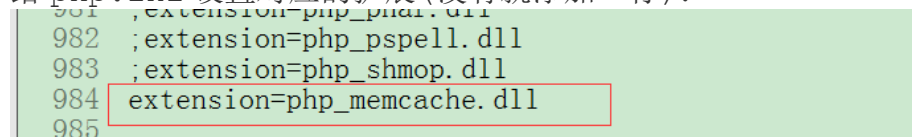


停止和卸载服务:

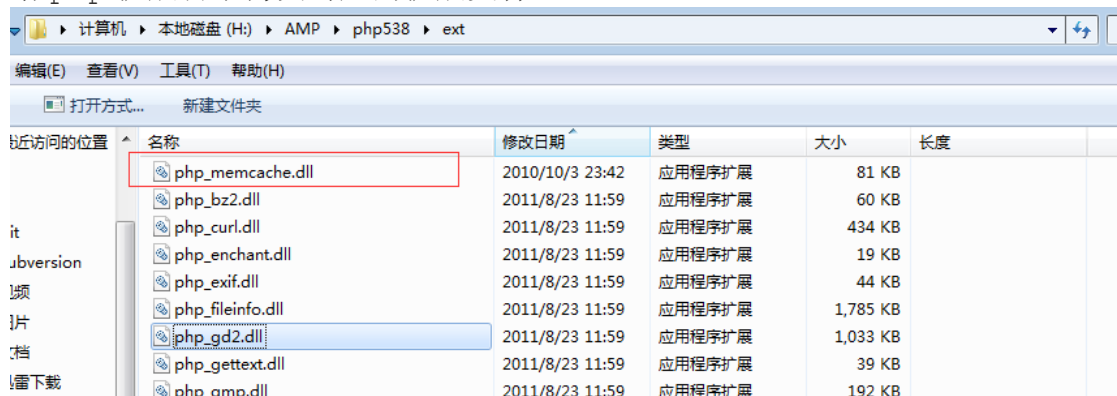


2.3 3. 给 php 开启 memcache 扩展

给 php.ini 设置对应的扩展 (没有就添加一行):

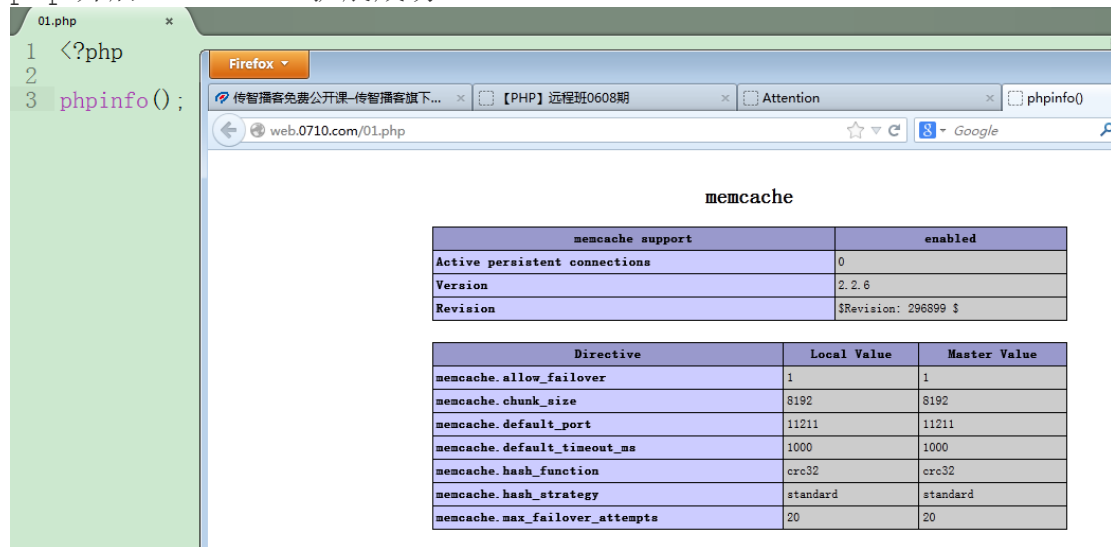


给 php 扩展目录拷贝对应的扩展文件:



之后重启 apache

php 开启 memcache 扩展成功:



2.4 Linux 下安装 memcached

参考网址: <https://github.com/memcached/memcached>
http://blog.csdn.net/sinat_21125451/article/details/50983343
<http://www.linuxidc.com/Linux/2015-05/117170.htm>

【 <http://www.centoscn.com/image-text/config/2016/0525/7248.html> 】

安装环境: centos 7, 安装包及顺序如下:

libevent-2.1.8-stable.tar.gz

libmemcached-1.0.18.tar.gz

memcached-1.4.36.tar.gz

memcached-3.0.3.tgz (连接 php 的插件, 通过 phpize 安装)

2.4.1 软件安装

1. 安装 memcached-1.4.36.tar.gz

注意依赖软件

2.4.2 php-memcached 扩展

memcached-3.0.3.tg.gz (为 memcached 的格式)

通过 phpize 安装 memcached-3.0.3.gz:

(1) 进入 memcached-3.0.3.tg.gz 解压目录, 执行 `/usr/local/php7/bin/phpize`

(2) `./configure--with-php-config=/usr/local/php/bin/php-config` 执行安装。

安装以上软件, 安装 memcached 之后, 会在以下目录下生成扩展文件:

```

[root@localhost no-debug-non-zts-20160303]# ls
memcached.so opcache.so redis.so
[root@localhost no-debug-non-zts-20160303]# pwd
/usr/local/php7/lib/php/extensions/no-debug-non-zts-20160303

```

(3) memcached.so 文件

进入 php.ini 中, 添加配置文件:

```

926 ;extension=php_xsl.dll
927 extension=redis.so
928 extension=memcached.so

```

通过 phpinfo() 查看会否已经配置好 memcached。

或者使用 php -m

2.4.3 php-memcache 扩展

<https://pecl.php.net/get/memcache-2.2.5.tgz>

2.4.4 启动 memcached 服务

(1) 目前有效的方式:

进入 root 身份, 输入以下命令启动 memcached

```

/usr/local/memcached/bin/memcached -b -l 127.0.0.1 -p
11211 -m 150 -u root

```

参数的说明:

-d 选项是启动一个守护进程,

-m 是分配给 Memcached 使用的内存数量, 单位是 MB, 我这里是 150MB,

-u 是运行 Memcached 的用户, 我这里是 root,

-l 是监听的服务器 IP 地址, 如果有多个地址的话, 我这里指定了服务器的 IP 地址 127.0.0.1,

-p 是设置 Memcached 监听的端口, 我这里设置了 11211, 最好是 1024 以上的端口,

-c 选项是最大运行的并发连接数, 默认是 1024, 我这里设置了 256, 按照你服务器的负载量来设定,

-P 是设置保存 Memcached 的 pid 文件, 我这里是保存在 /tmp/memcached.pid

(2) 查看是否启动:

查看线程 pid: ps -A | grep mem

```

[ncscherb@localhost ~]$ ps -A | grep mem
13846 pts/1    00:00:07 memcached

```

查看线程详细信息: ps auxxww | grep memcached

```

ps auxxww | grep memcached
[ncscherb@localhost ~]$ ps auxxww | grep memcached
root      13846  0.0  0.0 324724 1468 pts/1    S1+  09:17   0:06 ./memcached -b -l 127.0.0.1 -p 11211 -m 150 -u root
ncscherb  19125  0.0  0.0 112648 1008 pts/3    S+   11:47   0:00 grep --color=auto memcached

```

(3) 关闭 memcached 服务

杀死相应的线程即可: kill -9 13846

(4) 开机启动 memcached 服务

2.4.5 telnet 连接 memcached 服务

```
centos 安装 telnet
yum install telnet-server
yum install telnet
```

此时输入: telnet 127.0.0.1 11211 (为 memcached 设置的端口号), 显示信息如下:

```
[ncscherb@localhost ~]$ telnet 127.0.0.1 11211
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^'.
```

然后输入 stats, 显示信息如下, 表示 memcached 能正常连接和访问:

```
[ncscherb@localhost ~]$ telnet 127.0.0.1
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^'.
stats
STAT pid 13846
STAT uptime 44
STAT time 1492305475
STAT version 1.4.36
STAT libevent 2.1.8-stable
STAT pointer_size 64
STAT rusage_user 0.000000
STAT rusage_system 0.076673
```

2.5 php 操作 memcached

2.5.1 php 连接 memcached

```
<?php
```

```
    $m = new Memcached(); //实例化 Memcached 类
    $arr = array(
        array('127.0.0.1', 11211)
    );
```

```
    $m->addServers($arr);
```

\$m->set('name', 'lsgogroup', 3600); //设置缓存值, 有效时间 3600 秒, 如果有效时间设置为 0, 则表示该缓存值永久存在的 (系统重启前)

```
    echo $m->get('name'); //读取缓存值
```

```
    $m->delete('name'); //删除缓存值
```


2.5.2 php 获取/设置 memcached

提问：memcached 与 memcache 的区别？

在 php 中 memcache 体现为“类 Memcached”

具体使用：实例化对象，对象调用成员方法即可。

具体操作：

设置

`$obj -> set(key, value, 是否有压缩 0/1, 有效期);`

是否压缩：

不考虑速度，计较内存空间，压缩

计较速度，不计较内存空间，不压缩

有效期：

单位：秒

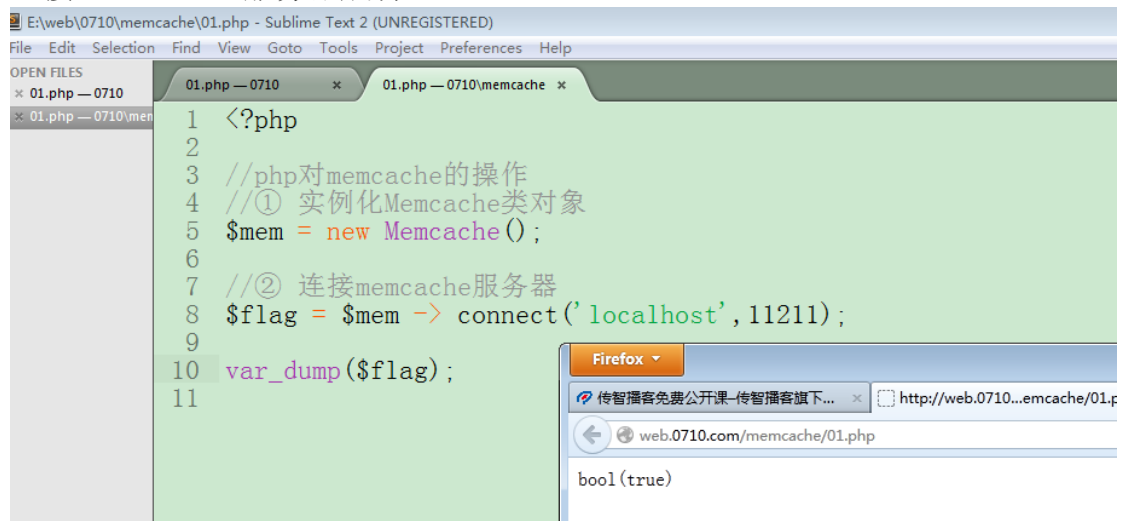
获取

`$obj -> get(key);`

删除

`$obj -> delete(key);`

连接 memcache 服务器成功：



给 memcache 设置一个 week 的 key 变量：

```
\web\0710\memcache\02.php - Sublime Text 2 (UNREGISTERED)
Edit Selection Find View Goto Tools Project Preferences Help

N FILES
1.php — 0710
1.php — 0710\men
2.php

01.php — 0710 x 01.php — 0710\memcache x 02.php x

1 <?php
2
3 //php对memcache的操作
4 //① 实例化Memcache类对象
5 $mem = new Memcache();
6
7 //② 连接memcache服务器
8 $flag = $mem -> connect('localhost', 11211);
9
10 //③ 给内存设置key
11 // $obj -> set(key, value, 是否有压缩, 有效期);
12 $mem -> set('week', 'Tuesday', 0, 3600*24);
13
14
```

把刚才设置好的 key 给读取出来：

```
0710\memcache\04.php - Sublime Text 2 (UNREGISTERED)
Selection Find View Goto Tools Project Preferences Help

0710
0710\men

01.php — 0710 x 01.php — 0710\memcache x 02.php x 04.php

1 <?php
2
3 //php对memcache的操作
4 //① 实例化Memcache类对象
5 $mem = new Memcache();
6
7 //② 连接memcache服务器
8 $flag = $mem -> connect('localhost', 11211);
9
10 //④ 把刚刚设置的key给读取出来
11 var_dump($mem -> get('week')); //string(7) "Tuesday"
12
13
```

Firefox

传智播客免费公开课-传智播客旗下... http://web.0710...

web.0710.com/memcache/04.php

string(7) "Tuesday"

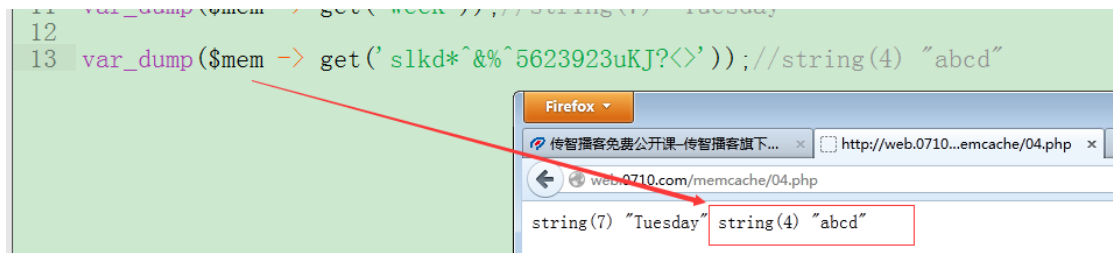
2.5.3 key 的命名

key 的名字可以有許多字符组成，长度不能超过 250 字节。

空格不能作为名字的组成内容。

(utf-8 字符集的一个汉字是 3 个字节)

```
13
14 //key的名字组成比较随意(可以有許多字符组成)
15 $mem -> set('slkd*^&%^5623923uKJ?<>', 'abcd', 0);
16
```



2.5.4 有效期

为 0 即不失效。

两种方式：

- ① 时间戳：1970-1-1 号到目前的秒数
- ② 时间差：时间数字，从目前往后延伸的时间长度
时间差的值大到一定程度与时间戳的值可以保存一致
(限制：时间差最多不能多于 30 天，否则其为时间戳)

两种方式设置 key 的有效期：

```
10 //③ 设置
11 $mem -> set('color', 'red', 0, 30); //时间差
12
13 $mem -> set('age', 23, 0, time()+30); //时间戳
```

获取有效期内的 key 信息：

```
10 //③ 读取
11 var_dump($mem -> get('color'));
12 var_dump($mem -> get('age'));
```

Firefox

http://web.0710...emcache/05.php

web.0710.com/memcache/05.php

string(3) "red" string(2) "23"

有效期的限制 (变形)：

```
13
14 //3600*24*30=2592000
15 $mem -> set('wea', 'sunshine', 0, 2591666); //时间差 (有效期近30天)
16 $mem -> set('weal', 'rain', 0, 2592789); //时间差(时间戳) (1970-1-31后)
17
```

前者的信息可以正常获取，后者已经早早过期

```
10 //③ 读取
11 var_dump($mem -> get('color'));
12 var_dump($mem -> get('age'));
13 var_dump($mem -> get('wea'));
14 var_dump($mem -> get('weal'));
```

Firefox

http://web.0710...emcache/05.php

web.0710.com/memcache/05.php

string(3) "red" string(2) "23" string(8) "sunshine" bool(false)

一个 key 的有效期为 60 天只能通过时间戳方式设置。

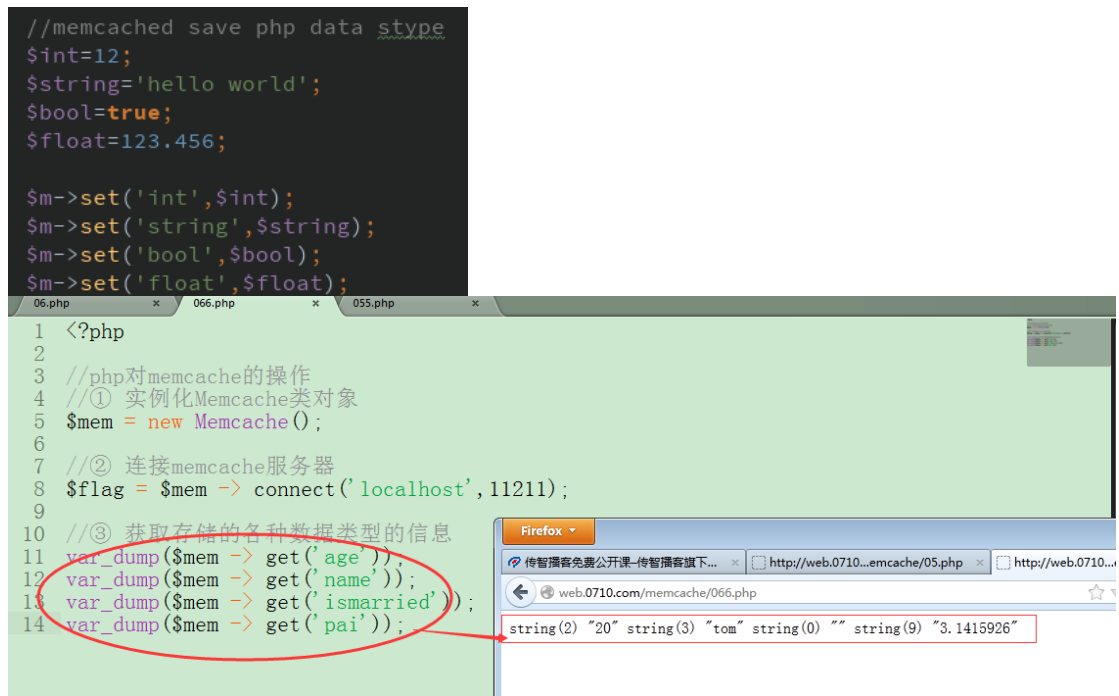
2.5.5 各种数据类型的存储

php 的数据类型 (8 种)：

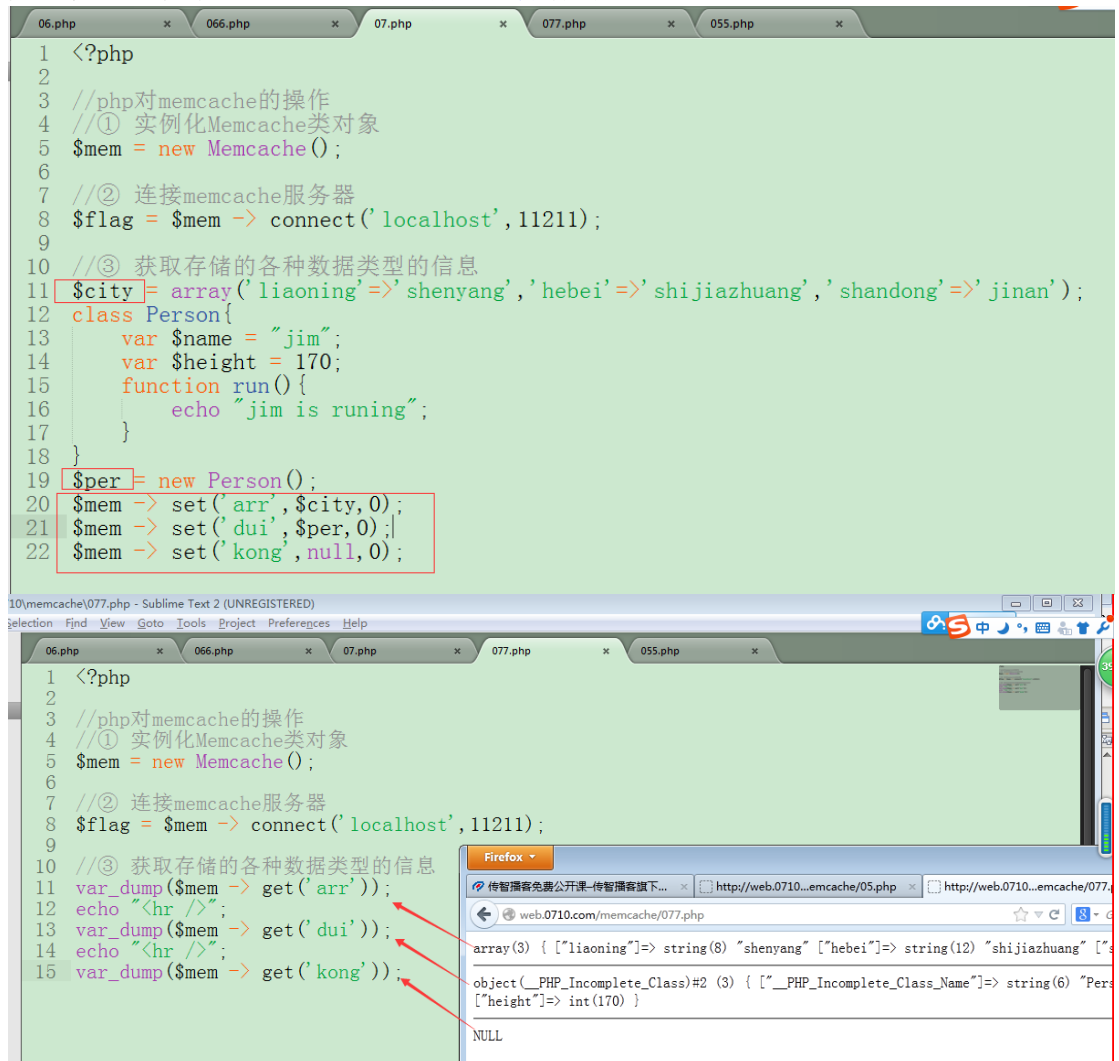
基本类型：int string boolean float

复合类型：array object resource null

基本类型 数据在 memcache 内部通过字符串存储，但是获取时仍然是原来的类型的值：



复合类型 数据在 memcache 中是原样存储:



有的时候在 memcache 中需要把各种数据类型信息都变为字符串存储，就需要对复合类型信息进行序列化操作：`serialize()` `unserialize()`

总结：

1. memcache

2. 安装、开启服务

3. php 中使用 memcache

设置：`set(key,value,0,有效期)`

获取：`get(key)`

删除：`delete(key)`

2.5.6 4.4 第三个参数压缩作用

通过 `zlib` 进行压缩处理

2.5.7 4.5 php 中其他相关操作方法

`add()` 给 memcache 增加一个 key

不存在就增加，存在就报错

`set()` 给 memcache 设置 key

不存在就增加，存在就修改

`close()` 关闭 memcache 连接，一般根据具体情况，设置到 php 代码的最后

`decrement()` 给 key 的值减少 1 `i--`

`increment()` 给 key 的值累加 1 `i++`

`flush()` 清空 memcache 的全部 key

`replace()` 替换 key 的值为其他值

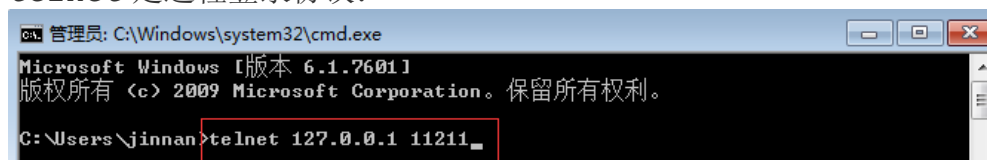
存在就替换，不存在就报错

3 终端命令方式操作

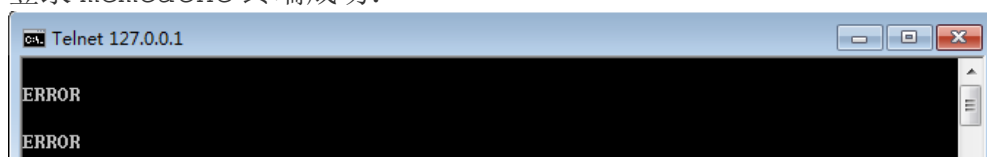
3.1 telnet 登录到 memcached 终端

登录到 memcache 的操作终端：

telnet 是远程登录协议：

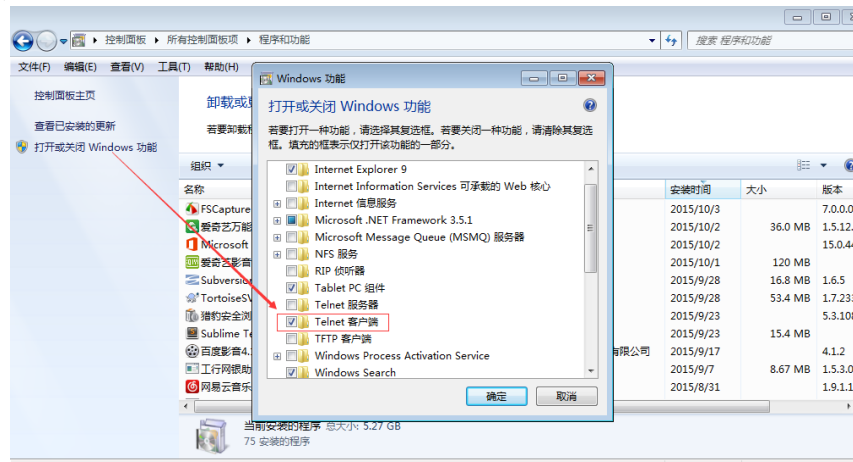


登录 memcache 终端成功：



telnet 提示不是内部或外部指令的解决:

控制面板--》程序和功能--》打开或关闭 windows 服务--》telnet 客户端:



3.2 在终端窗口实现 memcache 的操作

3.2.1 终端获取设置 memcached

> set key 是否压缩 有效期 数据长度 [回车]
> 数据

> add key 是否压缩 有效期 数据长度 [回车]
> 数据

> replace key 是否压缩 有效期 数据长度 [回车]
> 数据
(数据真实长度 与 设置长度 要完全一致)

获取:

> get key

删除:

> delete key

> flush_all //删除全部的 key

给 memcache 设置一个 key 和读取:

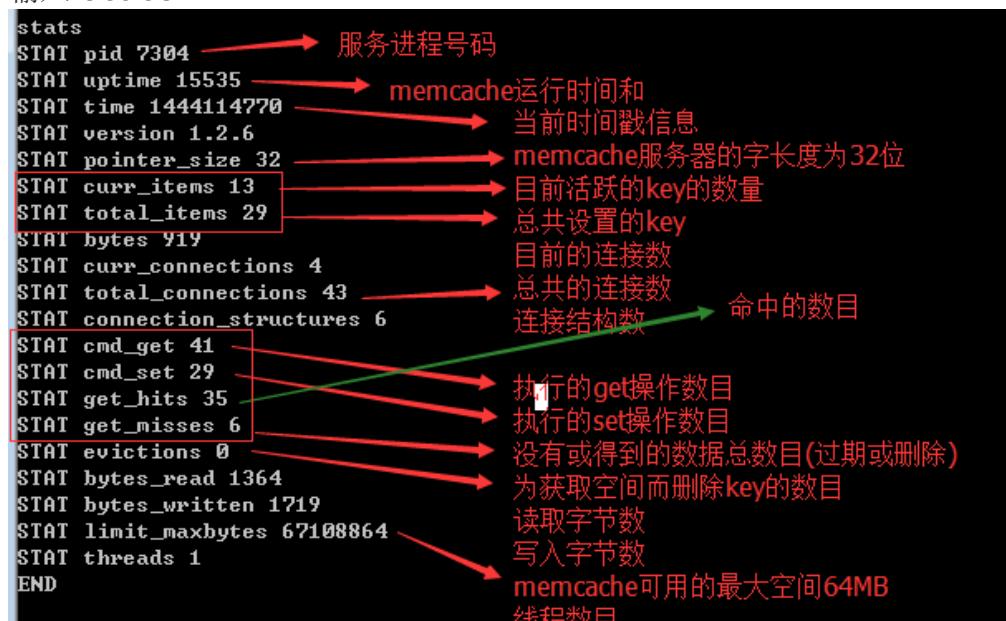
```
set addr 0 500 ?  
beijing  
STORED  
get addr  
VALUE addr 0 ?  
beijing  
END
```

通过 php 程序也可以读取到终端窗口设置的 key:



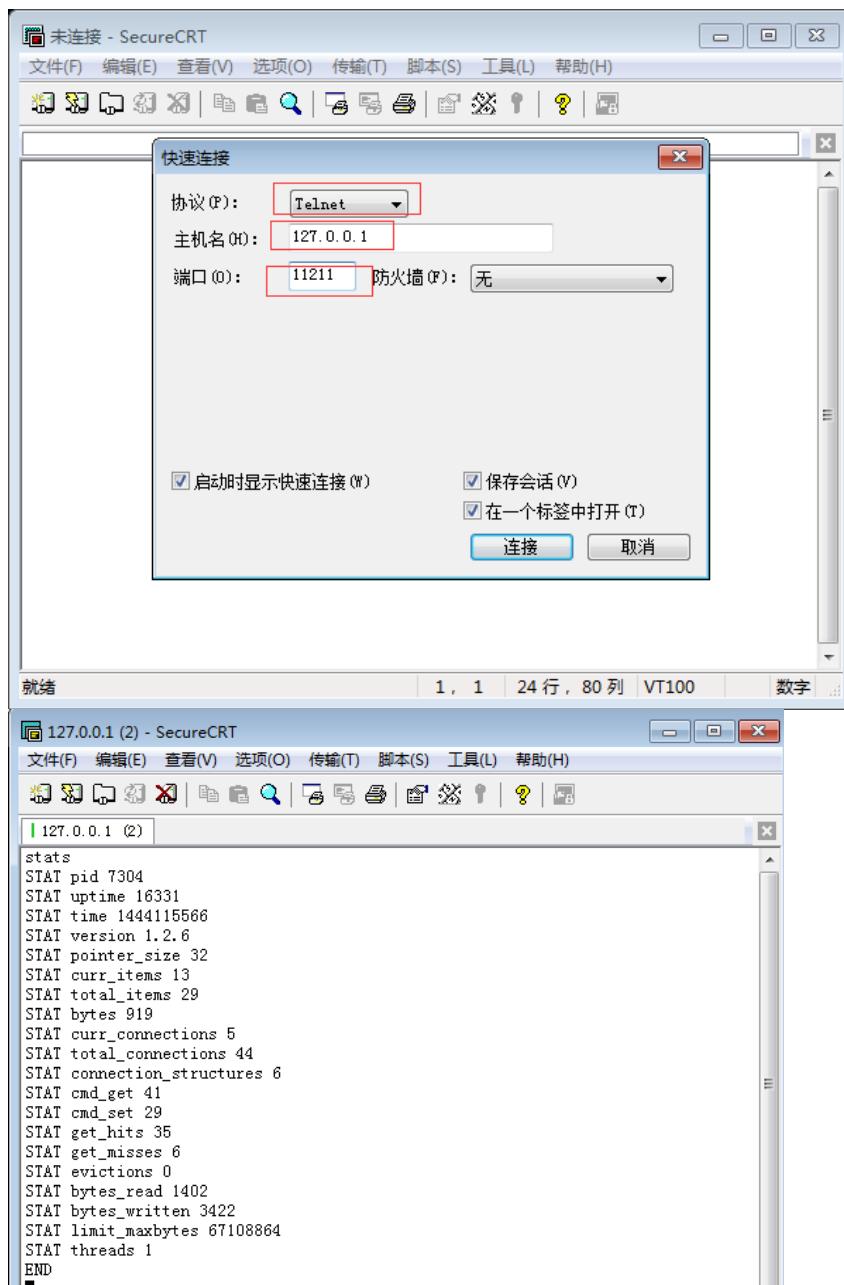
3.2.2 获取 memcache 统计的信息

输入 stats



在 php 程序里边可以通过 `getStats()` 获得 memcache 服务器的统计信息。

利用 SecureCRT 也可以实现对 memcache 的终端操作：



3.3 6. 分布式 memcache 的部署

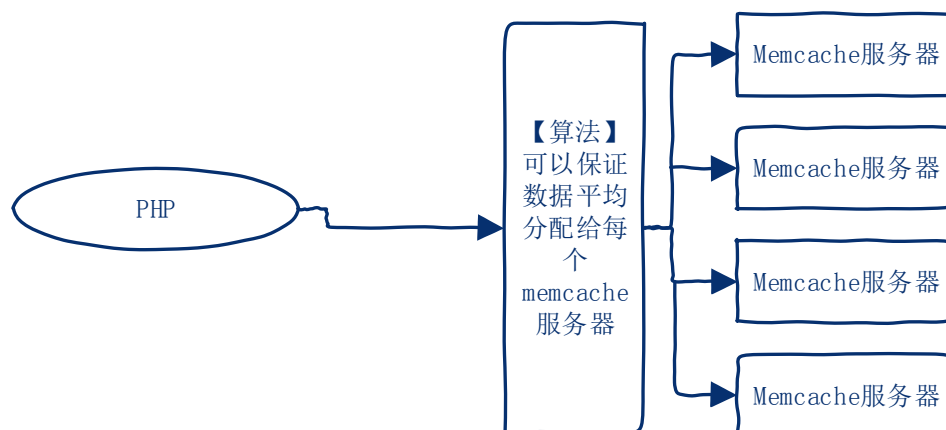
如果单个 memcache 保存的数据非常多，memcache 本身工作负载就会非常高，为了降低该 memcache 的工作量，提高其运行速度，可以设置多个 memcache 平均分担工作量，该模式就是[分布式](#)。

(例如一个 memcache 要保存 1000W 的数据，如果平均分配到 5 个 memcache 服务器，则每个就只保留 200W 的数据)

Redis 的分布式是“主从模式”结构，一主多从。
Memcache 的分布式与 Redis 的不同，其是把一台 memcache 的工作平均分配给多个 memcache 分担。

分布式具体的实施：

- 1) 可以在一个服务器里边开启多个 memcache 服务
- 2) 可以配置多个服务器，每个服务器里边都运行 memcache 服务



每个 memcache 服务器都是平等的，中间通过“算法”保证数据的平均分配。

php 代码的编写还保持原有习惯即可。

key 的分配原则：依次轮询、求余

一台服务器开启三个 memcache 服务：

The screenshot displays three command prompts running the command `memcached -p 11211`, `memcached -p 11212`, and `memcached -p 11213`. Red boxes highlight the command prompts and the corresponding processes in the Windows Task Manager window. The Task Manager window shows the following processes:

映像名称	PID	用户名
SoftMgrLite.exe *32	4652	jinnan
SGTool.exe *32	880	jinnan
SecureCRT.EXE *32	1240	jinnan
FSpider.exe *32	8536	jinnan
REDAgent.exe *32	7944	jinnan
QQ.exe *32	6628	jinnan
NvxDsync.exe	1152	SYSTEM
nvsvnc.exe	1160	SYSTEM
mmc.exe	7952	jinnan
memcached.exe *32	7424	jinnan
memcached.exe *32	6128	jinnan
memcached.exe *32	4452	jinnan
ICBCEBankAssist.exe	1876	jinnan
hh.exe	1812	jinnan
FSRecorder.exe *32	3432	jinnan
FSMousePlugin.exe *32	1796	jinnan

分布式 php 代码的设计：

```
06.php x 09.php x 099.php x 066.php x 07.php
1 <?php
2
3 //php对memcache的操作
4 //① 实例化Memcache类对象
5 $mem = new Memcache();
6
7 //② 连接memcache服务器(分布式)
8 $mem -> addServer('127.0.0.1', 11211);
9 $mem -> addServer('127.0.0.1', 11212);
10 $mem -> addServer('127.0.0.1', 11213);
11
12 //③ 设置key
13 $mem -> set('city1', 'beijing', 0);
14 $mem -> set('city2', 'shanghai', 0);
15 $mem -> set('city3', 'guangzhou', 0);
16
```

无需考虑 key 存储在那个 memcache 服务器内部，memcache 通过算法会自动给匹配上，不影响我们正常获得数据：

```
06.php x 09.php x 099.php x 066.php x 07.php x 077.php x 088.php x 055.php
1 <?php
2
3 //php对memcache的操作
4 //① 实例化Memcache类对象
5 $mem = new Memcache();
6
7 //② 连接memcache服务器(分布式)
8 $mem -> addServer('127.0.0.1', 11211);
9 $mem -> addServer('127.0.0.1', 11212);
10 $mem -> addServer('127.0.0.1', 11213);
11
12 //③ 设置key
13 var_dump($mem -> get('city1'));
14 var_dump($mem -> get('city2'));
15 var_dump($mem -> get('city3'));
16
```

Firefox

Attention x http://web.0710...emcache/099.php x +

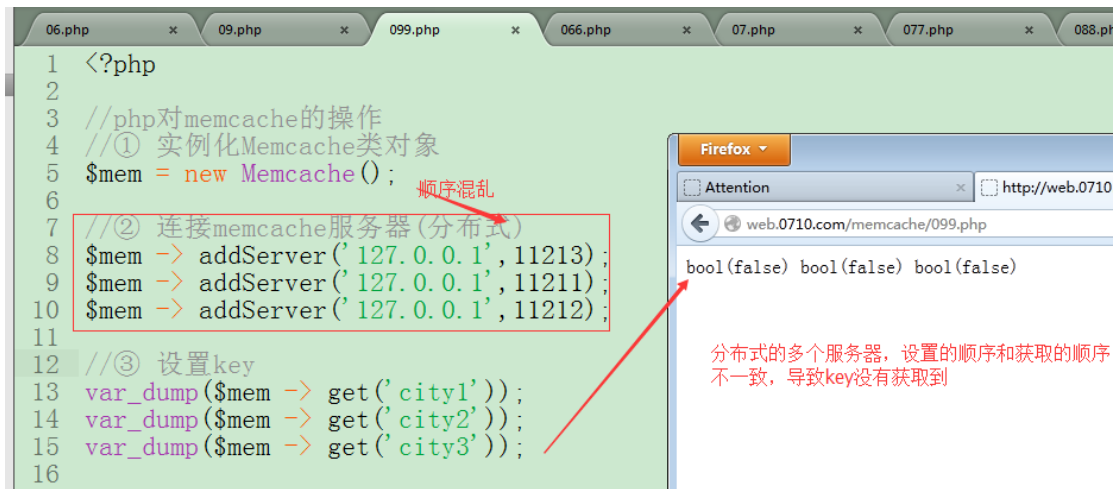
web.0710.com/memcache/099.php

string(7) "beijing" string(8) "shanghai" string(9) "guangzhou"

注意：

在每个 php 脚本文件内部服务器连接的顺序都要保持一致，否则数据有可能获取不到。

```
6
7 //② 连接memcache服务器(分布式)
8 $mem -> addServer('127.0.0.1', 11211);
9 $mem -> addServer('127.0.0.1', 11212);
10 $mem -> addServer('127.0.0.1', 11213);
11
```



3.4 7. 缓存失效

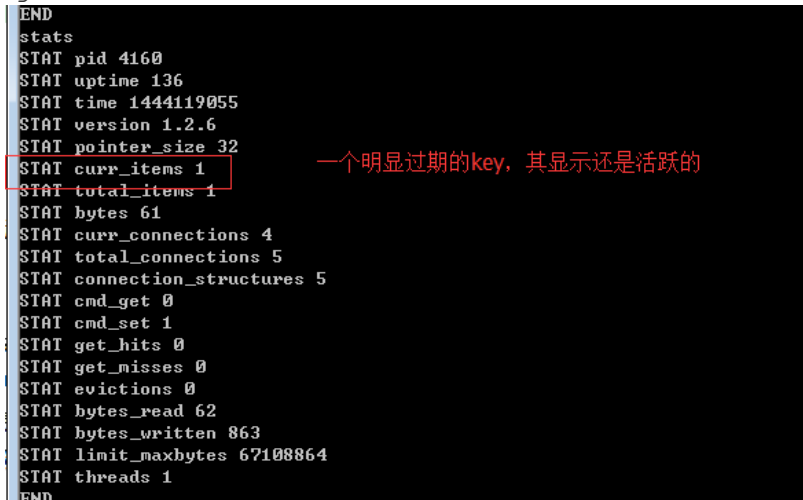
memcache 中的 key 超过有效期、或被系统强制删除掉了。

3.4.1 7.1 有效期过期

session 信息过期 (失效了)，通过“**懒惰**”模式给删除的。

session 是在文件中存储，如果 session 已经过期，其文件还是存在的，下次有一个用户访问 session 信息 (用户登录系统)，此时已经过期的 session 就有一定的几率被删除 (session 文件被删除)。

memcache 中 key 的删除也是**懒惰模式**，如果超过有效期，该 key 还是存在的，当你 get 获取它的时候，其就消失了。



```

get school
END
stats
ERROR
stats
STAT pid 4160
STAT uptime 193
STAT time 1444119112
STAT version 1.2.6
STAT pointer_size 32
STAT curr_items 0
STAT total_items 1
STAT bytes 0
STAT curr_connections 4
STAT total_connections 5
STAT connection_structures 5
STAT read_get 1

```

获取一个过期的key，就通过懒惰模式删除了

3.4.2 7.2 空间不足被强制删除

memcache 的内存可用空间默认为 64MB，如果存储的数据非常多，可用空间不足了。

此时仍然可以存储数据，因为 memcache 内部有 LRU 机制。

LRU: Least recently use 最近很少被使用的数据
内存空间如果不足，就会删除最近很少经常使用的数据。

如果不想使用 LRU 机制，就可以设置参数-M

开启 memcache 服务的时候带参数-M:

```

H:\>memcached -help
memcached 1.2.6
-p <num>      TCP port number to listen on (default: 11211)
-U <num>      UDP port number to listen on (default: 0, off)
-s <file>     unix socket path to listen on (disables network support)
-a <mask>     access mask for unix socket, in octal (default 0700)
-l <ip_addr>  interface to listen on, default is INADDR_ANY
-d start      tell memcached to start
-d restart    tell running memcached to do a graceful restart
-d stop!shutdown tell running memcached to shutdown
-d install    install memcached service
-d uninstall  uninstall memcached service
-r            maximize core file limit
-u <username> assume identity of <username> (only when run as root)
-m <num>     max memory to use for items in megabytes, default is 64 MB
-M           return error on memory exhausted (rather than removing items)
-c <num>     max simultaneous connections, default is 1024
-k           lock down all paged memory. Note that there is a
            limit on how much memory you may lock. Trying to

```

-M: 内存空间耗尽，要报错，而不使用 LRU 机制删除数据

3.5 8. session 入 memcache

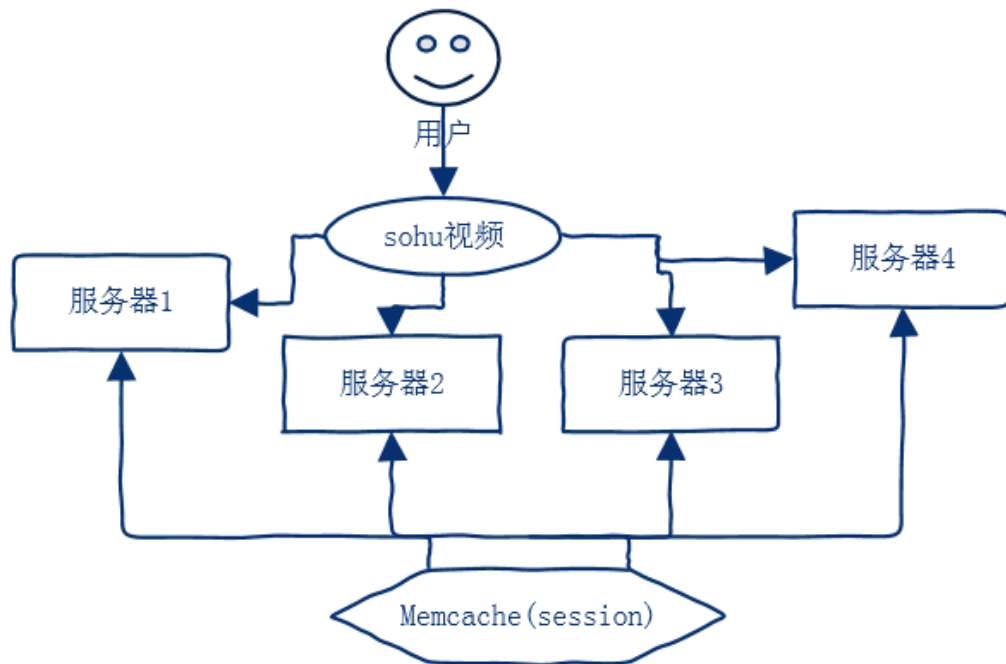
传统 session 的数据是在硬盘的文件中存储的。

该 session 很大情况用于存储用户的相关信息。用于判断一个用户是否登录系统。

两个服务器的 session 如果是文件形成存储，则他们的 session 互相不能通信。

两个服务器的 session 如果是存储在 memcache 中的，则他们的 session 可以通信。

内容分布式部署



一个网站是有多个服务器支撑的，用户在服务器 1 里边登录系统，其 session 持久化的信息报保存在一个 memcache 服务器里边，这样服务器 2/3/4 也可以去 memcache 读取 session 信息，就可以保证用户访问各个服务器的时候无需重复登录系统。

(以上情况还可以把 session 存储在 mysql 中)

3.5.18.1 具体操作

php.ini 关于 session 的设置：

存储 session 形式：

```
1449
1450 [Session]
1451 ; Handler used to store/retrieve data.
1452 ; http://php.net/session.save-handler
1453 session.save_handler = files
1454
```

存储在哪：

```
1479 ; where MODE is the octal representation of the mode. Note that this
1480 ; does not overwrite the process's umask.
1481 ; http://php.net/session.save-path
1482 ;session.save_path = "/tmp"
1483 session.save_path = H:/amp/php538/tmp/
1484
```

session 存储到 memcache 的设置：

```
10.php x 101.php x php.ini x
1 <?php
2
3 //实现session在memcache中存储
4 ini_set("session.save_handler","memcache");
5 ini_set("session.save_path","tcp://127.0.0.1:11211");
6
7
8 //正常操作session
9 session_start();
10 $_SESSION['username'] = "xiaoming";
11
12
```

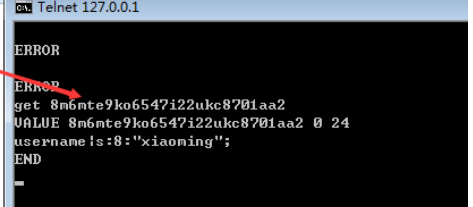
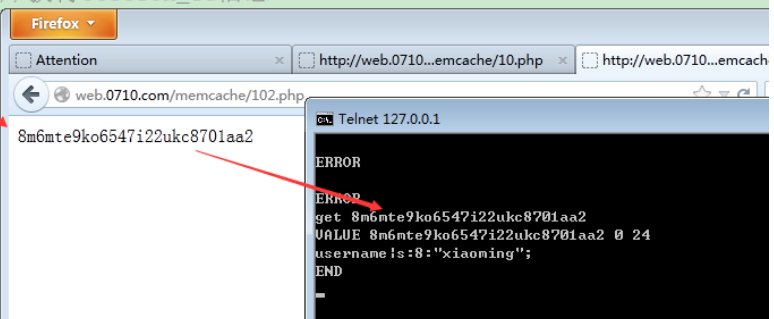
读取 memcache 中的 session 信息:

```
10.php x 101.php x php.ini x
1 <?php
2
3 //实现session在memcache中存储
4 ini_set("session.save_handler","memcache");
5 ini_set("session.save_path","tcp://127.0.0.1:11211");
6
7
8 //正常操作session
9 session_start();
10 var_dump($_SESSION['username']);
11
12
```



session 保存到 memcache 中 key 的名称为 `session_id` 的值:

```
10.php x 101.php x 102.php x php.ini x
1 <?php
2
3 //实现session在memcache中存储
4 ini_set("session.save_handler","memcache");
5 ini_set("session.save_path","tcp://127.0.0.1:11211");
6
7
8 //正常操作session
9 session_start();
10 echo session_id(); //获得session_id信息
11
12
```



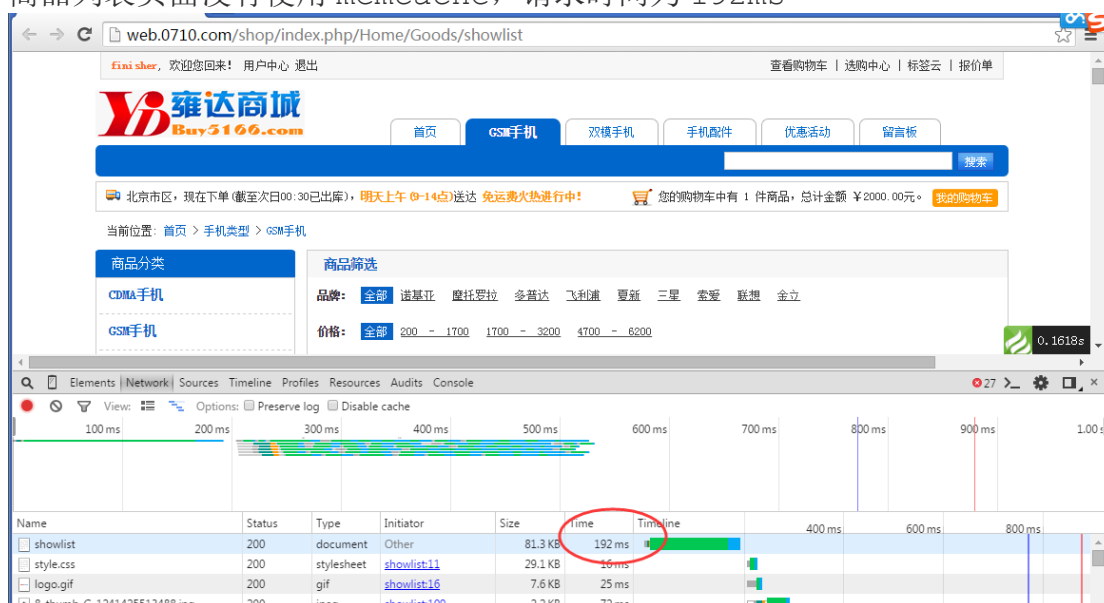
如果 memcache 是分布式的:

```
1 <?php
2
3 //实现session在memcache中存储
4 ini_set("session.save_handler", "memcache");
5 ini_set("session.save_path", "tcp://127.0.0.1:11211;tcp://127.0.0.1:11212;tcp://127.0.0.1:11213");
6
7
8 //正常操作
```

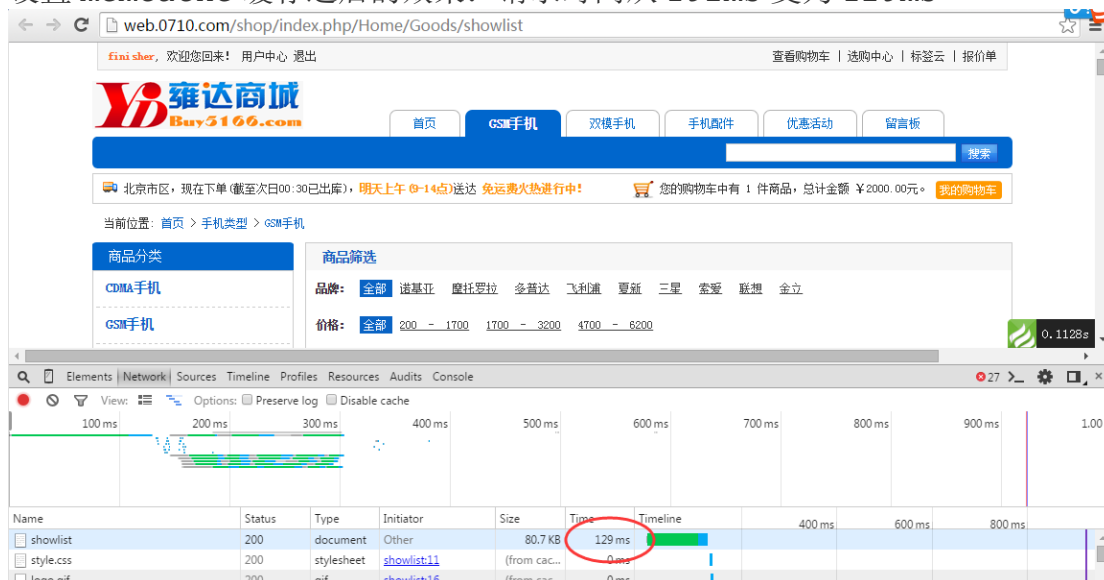
3.6 9. 案例效果

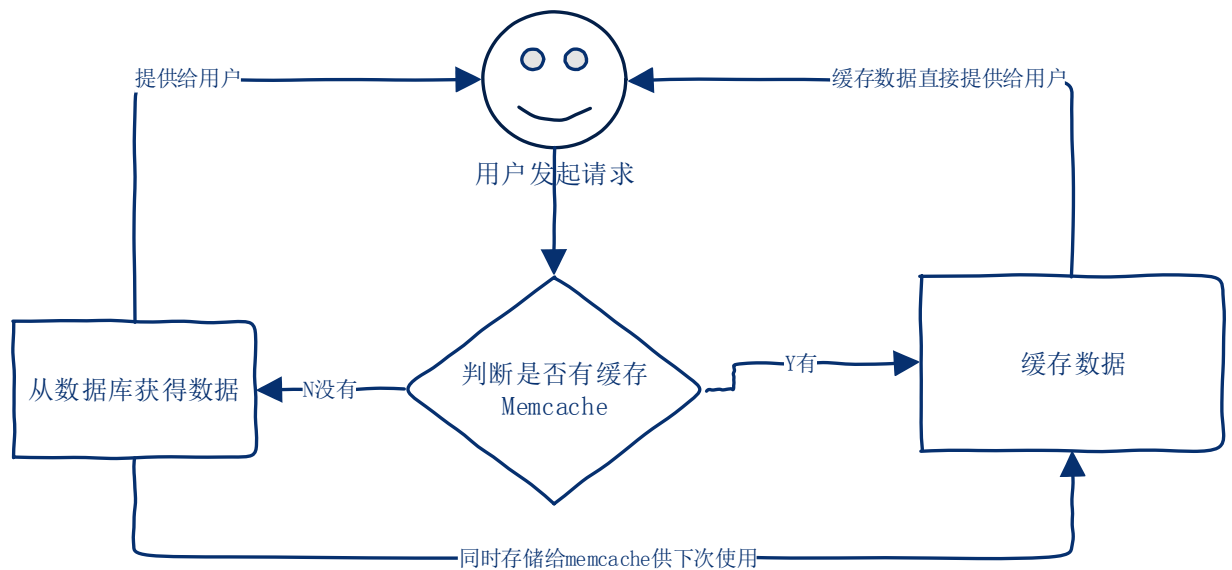
在网站前台商品列表页面处, 给商品列表信息存储在 memcache 中, 这样许多人在访问的时候就通过 memcache 获得数据, 提供页面的请求速度。

商品列表页面没有使用 memcache, 请求时间为 192ms



设置 memcache 缓存之后的效果: 请求时间从 192ms 变为 129ms





在 thinkphp 框架中使用 memcache 缓存:

`S()` 函数进行缓存的操作

通过 `S()` 函数连接 memcache 服务

`S(array('type'=>'memcache', 'host'=>主机名, 'port'=>端口号码))`

操作 key

`S(key, value, 有效期);`

获取 key

`S(key);`

删除 key

`S(key, null);`

```

9  function showlist() {
10     //设置memcache缓存
11     S(array('type'=>'memcache', 'host'=>'127.0.0.1', 'port'=>11211));
12
13     //给存储的商品信息设置一个名字: goods_category_info
14     //线上产品给key其名字, 是: md5(sql语句)
15     $info = S('goods_category_info');
16     if(empty($info)) {
17         echo "此时是mysql数据在发挥影响";
18         //获得商品列表信息
19         $info = D('Goods')->field('goods_name, goods_price, goods_small_img')->select();
20         //把获得好的信息存储在memcache中
21         S('goods_category_info', $info, 0);
22     }
23     //传递给模板
24     $this->assign('info', $info);
25     $this->display();
26 }
  
```

数据库的数据如果有做修改,就要删除旧的缓存,根据新的数据内容生成一个新缓存。

具体有两种实现方式:

```
217 function upd($goods_id){
218     //获得被修改的商品信息
219     //find() 获得数据表记录信息,每次通过“一维数组”返回一个记录结果
220     //model对象->find(); 获得第一个记录结果
221     //model对象->find(数字); 获得“主键id值”等于数字条件的记录结果
222     $goods = D('Goods');
223     //两个逻辑:展示、收集
224     if(!empty($_POST)){
225         $z = $goods->save($_POST);
226         //① 获得全部的商品信息,给前台页面制作一个新缓存删除
227         S(array('type'=>'memcache','host'=>'127.0.0.1','port'=>11211));
228         $info = D('Goods')->field('goods_name,goods_price,goods_small_img')->select();
229         S('goods_category_info',$info,0);
230         //② 或者直接删除旧数据缓存即可
231         S('goods_category_info',null);
232     }
233     if($z){
```

总结:

1. 终端方式操作 memcache

走 telnet 协议 (SecureCRT 也可以操作)

2. session 存入 memcache 中

多个服务器在真实环境中需要共享 session 信息,所以要存储 memcache 中

3. 缓存失效

1) 有效时间过期,通过懒惰模式删除

2) 空间不足,通过 LRU 方式强制删除最近不使用的 key

-M 参数可以禁止 LRU 的使用

4. 分布式部署 memcache

不同于主从模式,其为平均存储各个 key 到 memcache 中。

addServer (主机名, 端口);

5. 案例效果使用

作业:

1. 在项目里边实现 memcache 的应用。

2. 在自己的小组里边设计一个分布式部署的架构。

(有多个服务器,每个服务器都有 memcache)

```
checking for inttypes.h... yes
checking for stdint.h... yes
checking for unistd.h... yes
checking sasl/sasl.h usability... no
checking sasl/sasl.h presence... no
checking for sasl/sasl.h... no
configure: error: no, sasl.h is not available. Run configure with --disable-memcached-sasl to disable this check
[root@localhost memcached-3.0.3]#
```

```
./configure --enable-memcached
--with-php-config=/usr/local/php7/bin/php-config
--with-libmemcached-dir=/usr/local/libmemcached
```

--disable-memcached-sasl

```
[root@localhost memcached-3.0.3]# make install
Installing shared extensions:      /usr/local/php7/lib/php/extensions/no-debug-no
n-zts-20160303/
[root@localhost memcached-3.0.3]#
```