# Memcached

memcache 是分布式、高速、内存缓存服务器。<http://memcached.org/>

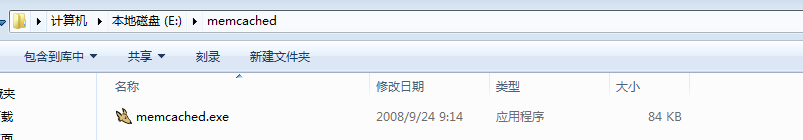
## 安装

Windows:

1.下载二进制运行程序



2.解压，拷贝到指定位置即可（Memcached.exe 就是memcached服务器）

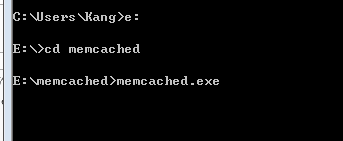


### 服务器端Memcached

**如何开启服务器**

**方式一：直接双击运行即可（基于命令行的执行型程序）**

**方式二：通过CMD进行管理**



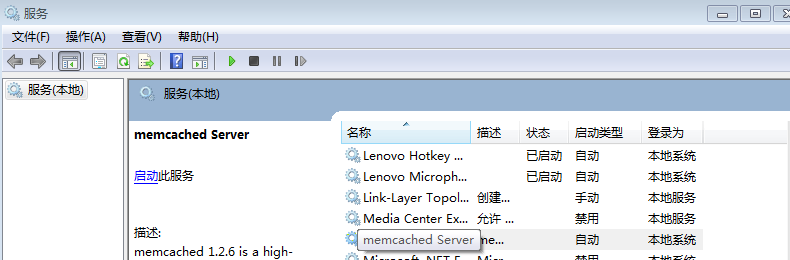
**方式三：添加到Windows服务(memcached –d install)**

* **需要以管理员身份运行 cmd**



**添加服务：**





**卸载服务：**



**建议在开发阶段选择第cmd 方式，生产阶段使用添加服务方式**

**Memcached管理命令**

memcached –h 获取帮助信息

开启服务器重要的选项：

memcached -p 监听端口 默认11211

memcached -l 监听IP

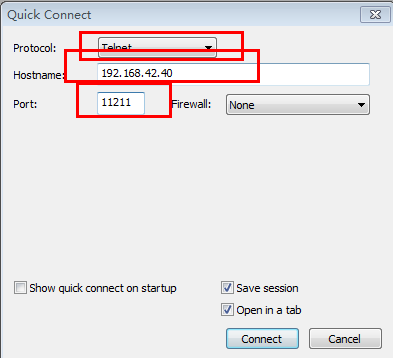
memcached -m 可占用最大的内存，以M兆为单位 默认64M

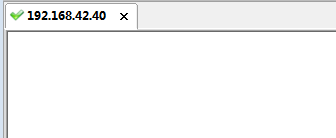


### 客户端

Cmd 等支持网络连接、可以发送文本指令的工具，就可以作为memcached 的客户端！

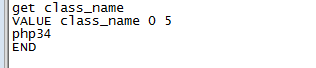
**如：SecureCRT** 选择以telnet协议进行远程连接





**客户端向服务器发送命令即可获得操作结果，如**





### Key/Value型数据存储

## 设置set

**set** *KEY 压缩标识 有效期 长度 VALUE*



压缩标识： 告知memcached服务器，是否压缩后存储数据。

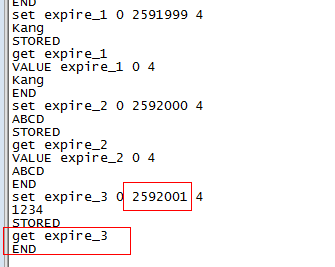
压缩目的为节省磁盘空间，但压缩与解压缩会消耗时间。

有效期： 用于表示KEY在什么时间失效。设置方式有两种，时间间隔和时间戳。

区分时间间隔和时间戳的办法：临界值 **30\*24\*3600=2592000**（30天的时间间隔）

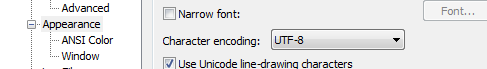
<=2592000 时间间隔

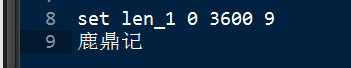
>2592000 时间戳



**通用的做法就是使用时间戳**

长度：值的大小，字节为单位（注意：utf-8 一个中文占三个字节）





键：字母数字下划线，标识符组成，长度不要大于250字节。

**实际开发时，建议使用前缀。**



值：需要缓存的数据内容，字符串，单个值不能超过1M大小！

**实际中，配合序列化与反序列化使用！**

**Tip: 不分增加和修改**

## 获取 get

get kEY



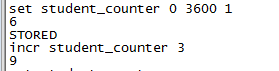
## 其他命令：incr，decr，add，replace，delete，flush\_all

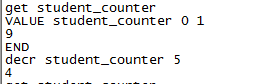
incr: 自增多少

decr: 自减多少

incr KEY N

decr KEY N

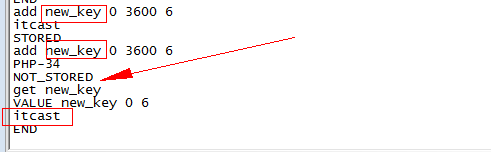


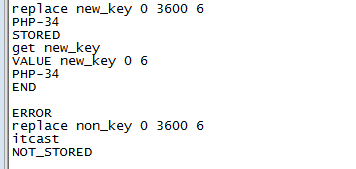


add： 仅添加，如果KEY已经，则添加**失败**

replace： 仅替换，如果KEY不存在，则替换**失败**

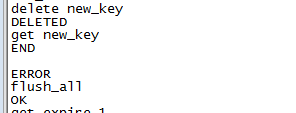
功能与set一致，限定了某些操作！





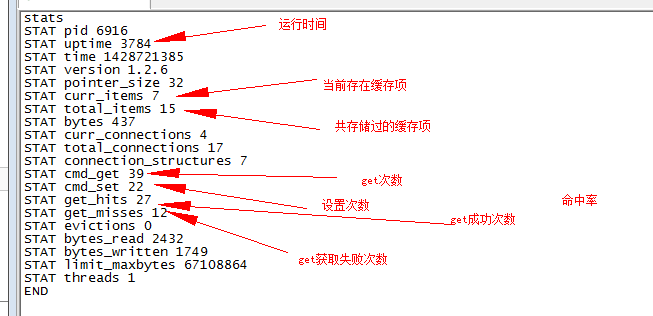
delete KEY 删除指定KEY缓存

flush\_all 清空所有缓存



## 状态 stats

获取当前memcached服务器的状态。



提升命中率，应该将，常用的，获取频繁的数据存储缓存中。

实际：缓存需要预热。填充缓存阶段。

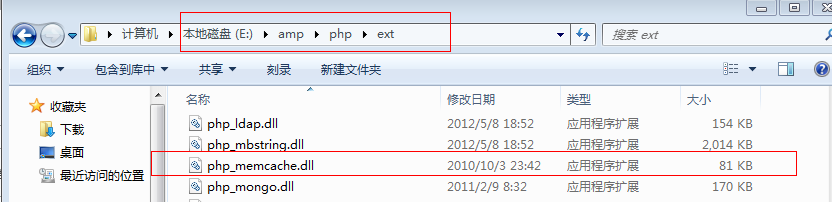
## PHP操作memcached

### 加载memcache扩展

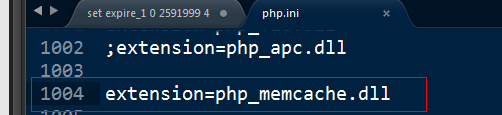
PHP 作为客户端存在！

开启memcache相关客户端扩展，php\_memcache.dll ：

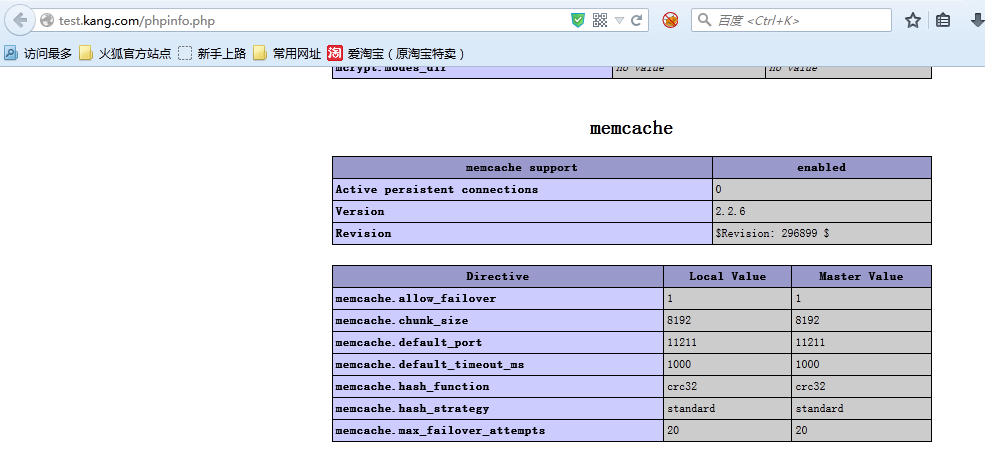
拷贝php\_memcache.dll文件，到PHP的扩展功能目录：



修改php.ini：

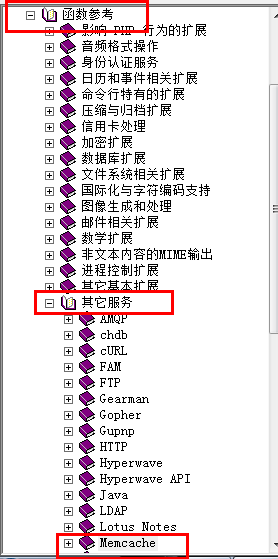


Apache httpd restart



提供OOP的语法，管理memcached服务器

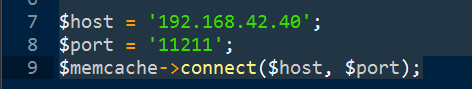
参考PHP手册中的memcache扩展函数部分：



### 实例化Memcache类



### 连接Memcached服务器



### 设置

$memcache->add()

$memcache->set(KEY, VALUE, 压缩标识, 有效期);

$memcache->replace();

$memcache->increment(KEY, N);

$memcache->decrement();



### 获取



### 删除

$memcache->delete(); 删除某个key

$memcache->flush(); 删除全部

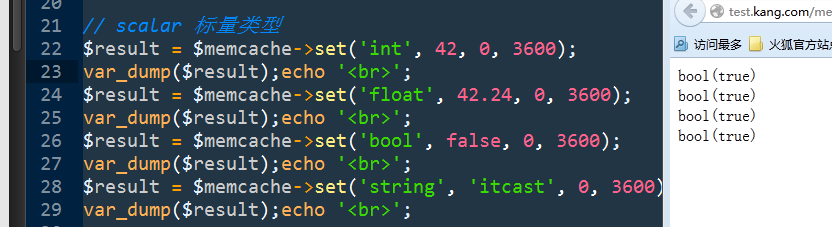


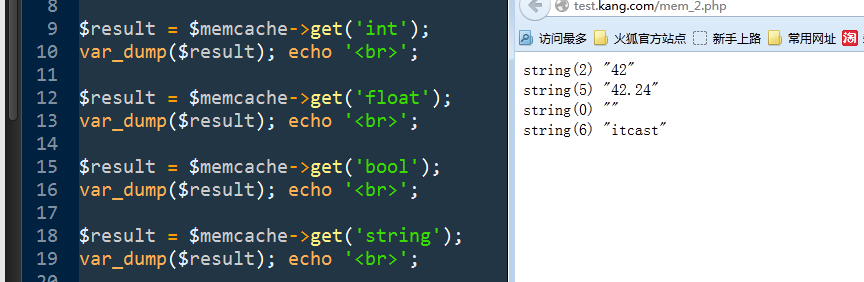
### 关闭连接



### 数据类型支持

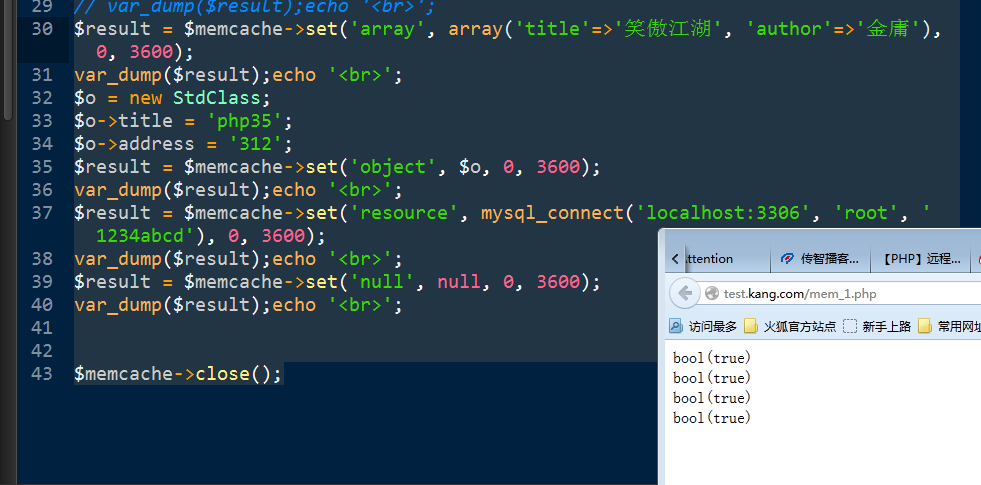
**标量类型：**

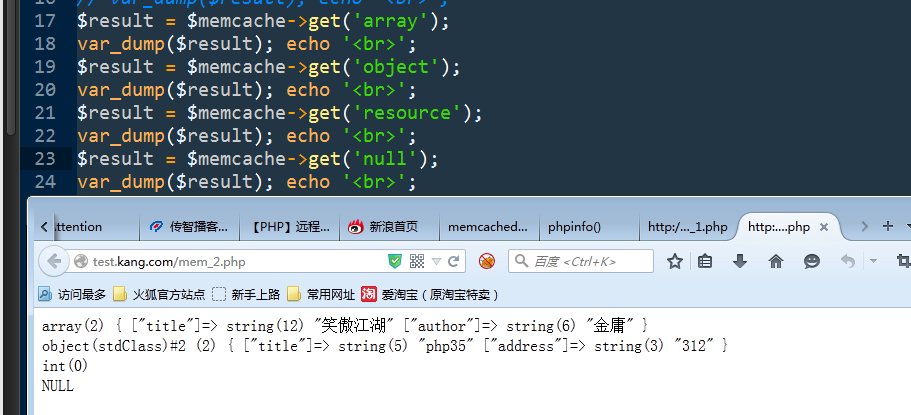




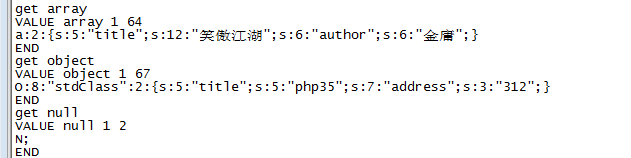
可见：如果是标量类型，则将其内容转换成字符串进行存储。

**非标量类型：**





可见：**非标量类型序列化存储，保持原有数据类型。**



Tip: is\_scalar(); 判断数据是否为标量类型

## 分布式缓存服务器

分布式 -- 使用多台memcached服务器，形成memcached集群。目的是提升memcache所能使用硬件资源数量。

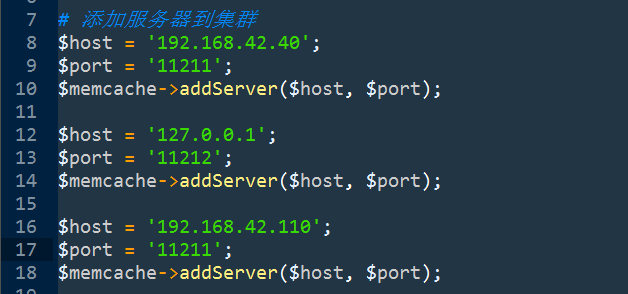
memcached 分布式是互不通信的memcached分布式。

多台memcached服务器间不相互通讯。指的是”分布式算法”由客户端来实现。（对PHP来说，memcache扩展来实现）



使用memcache对象的AddServer() 方法，可以添加多台memcached服务器到集群中。

透明的进行常规操作memcached 即可。如：set(), get(), delete()



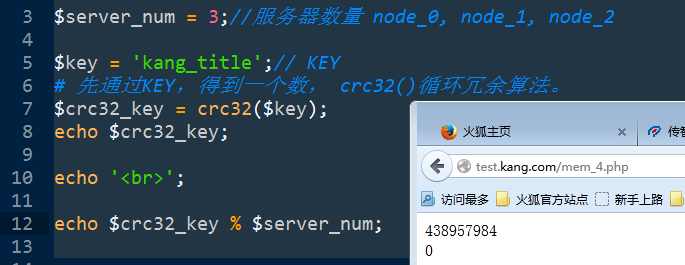


分布式算法：

求模

计算 KEY，通过keY与服务器数量求模。

典型的实现（不是memcache的完整实现，典型部分）：



由于memcached采用的也是 取模算法来确定key所对应的memcached服务器的。导致存取时对服务器的数量及其顺序有严格要求。

## 缓存失效

缓存过期

Memcached在处理过期的缓存项是，采用**懒惰模式**处理方法。

缓存项过期，不会立即删除，直到对该缓存项执行了***get操作***，才会删除过期的缓存项。

缓存空间已满

Memcache 在插入新数据时，如果空间不足采用的删除旧缓存项的策略。采用 删除 ***最近最少使用***（使用频率低）的缓存项（**RLU策略**）。

## Session入memcached

使用 **memcache扩展**提供的session处理器（session.save\_handler）即可。

通过存储位置配置项（session.save\_path）, 设置使用的memcached服务器信息。



此时，每个会话数据就是一个key/value型memcache的值，key 就是当前的session-id，值就是当前会话被序列号好的数据。



分布式的web服务器，也会促使将session入memcache或者入数据库：

