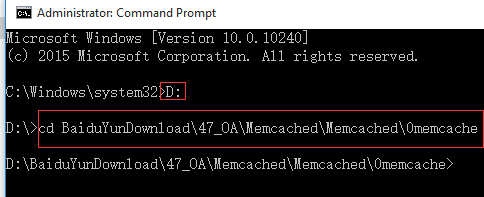
# 分布式缓存

## Memcached基本操作

Memcached操作路径中不要出现中文，Memcached以键值对的方式进行数据存储。

### 安装服务

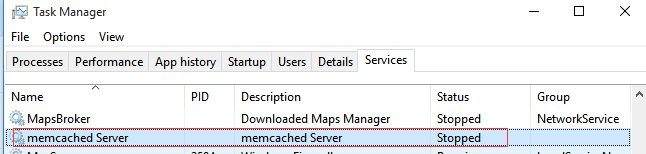
以管理员身份打开cmd，进入Memcached所在文件夹



安装服务（如果输入memcached.exe只是以程序打开，而不是服务）



在服务中，找到memcached Server。



打开服务



#### 补充：CMD命令

注意：要求以管理员身份进行操作

安装/卸载服务：memcached.exe -d install/uninstall

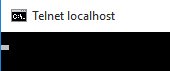
启动/关闭服务：memcached.exe -d start/stop

### 连接

使用telnet链接



连接成功



### Memached命令

命令语法

|  |
| --- |
| command <key> <flags> <expiration time> <bytes>  <value> |

|  |  |
| --- | --- |
| **参数** | **用法** |
| key | key 用于查找缓存值 |
| flags | 可以包括键值对的整型参数，客户机使用它存储关于键值对的额外信息 |
| expiration time | 在缓存中保存键值对的时间长度（以秒为单位，0 表示永远） |
| bytes | 在缓存中存储的字节点 |
| value | 存储的值（始终位于第二行） |

#### Set命令

set 命令用于向缓存添加新的键值对。如果键已经存在，则之前的值将被替换。

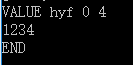


出现stored表示成功

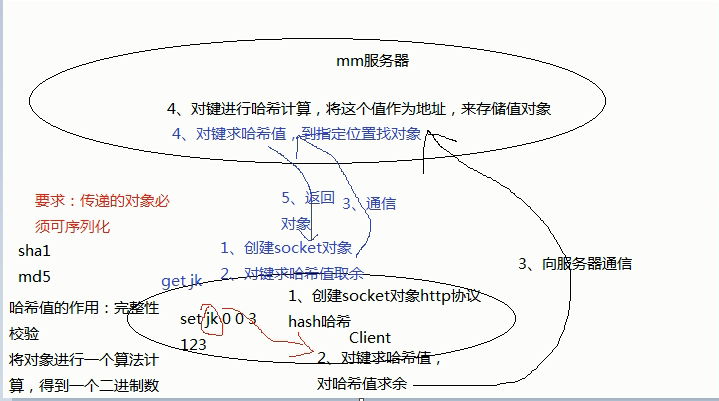
#### Get命令



结果：



## Memcached原理



当进行set时如

|  |
| --- |
| set hyf 0 0 3  123 |

实际进行了如下操作：

1. 创建socket对象。
2. 对键(hyf)求哈希值，然后对哈希值求余，得到的余数即为服务器的编号。
3. Socket对象通过http协议向服务器（上一步骤通过求余得到的服务器）通信。
4. 在mm服务器上，对键进行哈希值计算，将这个值作为内存地址，来存储值对象。

在整个通信过程中，进行了两次哈希值计算，第一次用来获取数据要存储的服务器的编号；第二次用来设置在服务器中的内存地址。

当进行get时，如

|  |
| --- |
| get hyf |

进行了如下操作：

1. 创建socket对象
2. 对键(hyf)求哈希值，然后对哈希值求余，得到的余数即为服务器的编号。
3. Socket对象通过http协议向服务器（上一步骤通过求余得到的服务器）通信。
4. 在mm服务器上，对键进行哈希值计算得到内存地址，从该内存地址中找到对象。
5. 返回对象。

在get set时 传递的对象必须可序列化。这样可以对二进制的对象添加校验信息，保证数据完整性。

## 哈希算法的缺陷

哈希算法存在投影不规则的问题。一方面对对象进行哈希值求值的结果是随机的，对哈希值求余后，有可能大量对象都映射到同一台服务器上，而其它服务器是空闲的。

此外，增加服务器时投影会发生改变。获取缓存对象时，与已存储了缓存对象的服务器不一致。例如，100台服务器时，求余后在第55台服务器，数据存在了这个服务器中，当服务器增加到110台后，求余可能对应的是第60台服务器，在获取对象时会第60台服务器取对象，而对象根本不存在。

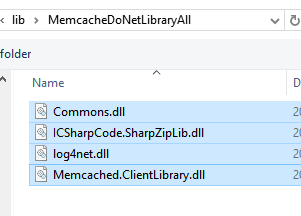
解决投影不规则与服务器数量扩展问题的方法是使用：一致性哈希算法。

一致性哈希算法对哈希值及求余进行了干预，解决增减主机、分布不均匀的问题。

* 哈希值求值：结果是0-232区间的不连续的值。
* 哈希值求余: 当服务器增加后求余时，大部分哈希值的投影不变，每一个服务器中只有小部分值会投射到新的服务器中，最后会达到所有服务器的负担均衡。服务器越多，投影变化会越小。

## Net中使用memcached

### 引入程序集



### 引入命名空间

|  |
| --- |
| using Memcached.ClientLibrary; |

### 编辑代码

|  |
| --- |
| static void Main(string[] args)  {  //1.配置服务器IP.服务器以逗号隔开{"127.0.0.1:11211","192.168.1.2:11211" }  string[] servers = { "127.0.0.1:11211" };  //2.初始化缓存池  SockIOPool pool = SockIOPool.GetInstance();  pool.SetServers(servers);  pool.Initialize();  //3.实例化对象进行操作.主要的方法Set，Add，Delete，Stats  MemcachedClient client = new MemcachedClient();  client.EnableCompression = true;  client.Set("hao", "name", 0);    Console.WriteLine(client.Get("hao").ToString());  Console.Read();  } |

可以用telnet进行get。表示成功

