



锁定老帖子 主题: [Memcached 学习笔记一](#)

该帖已经被评为良好帖

作者	正文
<div>leeqianjun</div> <div>等级: </div> <div></div> <div>性别: </div> <div>文章: 12</div> <div>积分: 220</div> <div>来自: 杭州</div> <div> 我现在离线</div>	<div>发表时间: 2008-11-07 最后修改: 2008-11-11</div> <div><div>&lt; &gt;</div> 猎头职位: <a href="#">北京: ITeye网站诚聘Ruby工程师</a></div> <div><h2>1 Memcache是什么</h2><p>Memcache是danga.com的一个项目,最早是为 LiveJournal 服务的,目前全世界不少人使用这个缓存项目来构建自己大负载的网站,来分担数据库的压力。</p><p>它可以应对任意多个连接,使用非阻塞的网络IO。由于它的工作机制是在内存中开辟一块空间,然后建立一个HashTable, Memcached自我管理这些HashTable。</p><p>为什么会有Memcache和memcached两种名称?</p><p>其实Memcache是这个项目的名称,而memcached是它服务器端的主程序文件名,</p><p>Memcache官方网站: <a href="http://www.danga.com/memcached">http://www.danga.com/memcached</a>,</p></div> <div><div>相关文章:</div><ul style="list-style-type: none"><li>▪ <a href="#">分布式缓存系统Memcached学习心得</a></li><li>▪ <a href="#">MemCached Cache Java Client封装优化历程</a></li><li>▪ <a href="#">基于memcached的SNA实现</a></li></ul><div>推荐群组: <a href="#">xmemached用户组</a> <a href="#">更多相关推荐</a></div></div> <div><h2>2 Memcache工作原理</h2><p>首先 memcached 是以守护程序方式运行于一个或多个服务器中,随时接受客户端的连接操作,客户端可以由各种语言编写,目前已知的客户端 API 包括 Perl/PHP/Python/Ruby/Java/C#/C 等等。客户端在与 memcached 服务建立连接之后,接下来的事情就是存取对象了,每个被存取的对象都有一个唯一的标识符 key,存取操作均通过这个 key 进行,保存到 memcached 中的对象实际上是放置内存中的,并不是保存在 cache 文件中的,这也是为什么 memcached 能够如此高效快速的原因。注意,这些对象并不是持久的,服务停止之后,里边的数据就会丢失。</p><p>与许多 cache 工具类似, Memcached 的原理并不复杂。它采用了C/S的模式,在 server 端启动服务进程,在启动时可以指定监听的ip,自己的端口号,所使用的内存大小等几个关键参数。一旦启动,服务就一直处于可用状态。Memcached 的目前版本是通过C实现,采用了单进程,单线程,异步I/O,基于事件(event_based)的服务方式.使用 libevent 作为事件通知实现。多个 Server 可以协同工作,但这些 Server 之间是没有任何通讯联系的,每个 Server 只是对自己的数据进行管理。Client 端通过指定 Server 端的 ip 地址(通过域名应该也可以)。需要缓存的对象或数据是以 key-&gt;value 对的形式保存在Server端。key 的值通过 hash 进行转换,根据 hash 值把 value 传递到对应的具体的某个 Server 上。当需要获取对象数据时,也根据 key 进行。首先对 key 进行 hash,通过获得的值可以确定它被保存在了哪台 Server 上,然后再向该 Server 发出请求。Client 端只需要知道保存 hash(key) 的值在哪台服务器上就可以了。</p><p>其实说到底, memcache 的工作就是在专门的机器的内存里维护一张巨大的 hash 表,来存储经常被读写的一些数组与文</p></div>

件，从而极大的提高网站的运行效率。

### 3 如何使用

#### · 建立Manager类


Java代码



```
1. package com.alisoft.sme.memcached;
2.
3. import java.util.Date;
4.
5. import com.danga.MemCached.MemCachedClient;
6. import com.danga.MemCached.SockIOPool;
7.
8. public class MemCachedManager {
9.
10.     // 创建全局的唯一实例
11.     protected static MemCachedClient mcc = new MemCachedClient();
12.
13.     protected static MemCachedManager memCachedManager = new MemCachedManager();
14.
15.     // 设置与缓存服务器的连接池
16.     static {
17.         // 服务器列表和其权重
18.         String[] servers = { "127.0.0.1:11211" };
19.         Integer[] weights = { 3 };
20.
21.         // 获取socket连接池的实例对象
22.         SockIOPool pool = SockIOPool.getInstance();
23.
24.         // 设置服务器信息
25.         pool.setServers(servers);
26.         pool.setWeights(weights);
27.
28.         // 设置初始连接数、最小和最大连接数以及最大处理时间
29.         pool.setInitConn(5);
30.         pool.setMinConn(5);
31.         pool.setMaxConn(250);
32.         pool.setMaxIdle(1000 * 60 * 60 * 6);
33.
34.         // 设置主线程的睡眠时间
35.         pool.setMaintSleep(30);
36.
37.         // 设置TCP的参数，连接超时等
38.         pool.setNagle(false);
39.         pool.setSocketTO(3000);
40.         pool.setSocketConnectTO(0);
41.
42.         // 初始化连接池
43.         pool.initialize();
44.
45.         // 压缩设置，超过指定大小（单位为K）的数据都会被压缩
46.         mcc.setCompressEnable(true);
47.         mcc.setCompressThreshold(64 * 1024);
48.     }
49.
50.     /**
```

```
51.         * 保护型构造方法，不允许实例化！
52.         *
53.         */
54.     protected MemCachedManager() {
55.
56.     }
57.
58.     /**
59.      * 获取唯一实例。
60.      *
61.      * @return
62.      */
63.     public static MemCachedManager getInstance() {
64.         return memCachedManager;
65.     }
66.
67.     /**
68.      * 添加一个指定的值到缓存中。
69.      *
70.      * @param key
71.      * @param value
72.      * @return
73.      */
74.     public boolean add(String key, Object value) {
75.         return mcc.add(key, value);
76.     }
77.
78.     public boolean add(String key, Object value, Date expiry) {
79.         return mcc.add(key, value, expiry);
80.     }
81.
82.     public boolean replace(String key, Object value) {
83.         return mcc.replace(key, value);
84.     }
85.
86.     public boolean replace(String key, Object value, Date expiry) {
87.         return mcc.replace(key, value, expiry);
88.     }
89.
90.     /**
91.      * 根据指定的关键字获取对象。
92.      *
93.      * @param key
94.      * @return
95.      */
96.     public Object get(String key) {
97.         return mcc.get(key);
98.     }
99.
100.     public static void main(String[] args) {
101.         MemCachedManager cache = MemCachedManager.getInstance();
102.         cache.add("hello", 234);
103.         System.out.print("get value : " + cache.get("hello"));
104.     }
105. }
```

# 建立数据对象

Java代码 

```
1. package com.alisoft.sme.memcached;
2.
```

```

3.  import java.io.Serializable;
4.
5.  public class TBean implements Serializable {
6.
7.      private static final long serialVersionUID = 1945562032261336919L;
8.
9.      private String name;
10.
11.     public String getName() {
12.         return name;
13.     }
14.
15.     public void setName(String name) {
16.         this.name = name;
17.     }
18. }

```

Java代码



```
1. <pre name="code" class="java"> </pre>
```

## 创建测试用例

Java代码



```

1.  package com.alisoft.sme.memcached.test;
2.
3.  import junit.framework.TestCase;
4.
5.  import org.junit.Test;
6.
7.  import com.alisoft.sme.memcached.MemCachedManager;
8.  import com.alisoft.sme.memcached.TBean;
9.
10. public class TestMemcached extends TestCase {
11.
12.     private static MemCachedManager cache;
13.
14.     @Test
15.     public void testCache() {
16.
17.         TBean tb = new TBean();
18.         tb.setName("E网打进");
19.         cache.add("bean", tb);
20.
21.         TBean tb1 = (TBean) cache.get("bean");
22.         System.out.println("name=" + tb1.getName());
23.         tb1.setName("E网打进_修改的");
24.
25.         tb1 = (TBean) cache.get("bean");
26.         System.out.println("name=" + tb1.getName());
27.     }
28.
29.     @Override
30.     protected void setUp() throws Exception {
31.         super.setUp();
32.         cache = MemCachedManager.getInstance();
33.     }
34.
35.     @Override
36.     protected void tearDown() throws Exception {
37.         super.tearDown();
38.         cache = null;

```

```
39.         }
40.
41.     }
```

测试结果

Java代码 ☆

```
1.  [INFO] ++++ serializing for key: bean for class: com.alisoft.sme.memcached.TBean
2.  [INFO] ++++ memcache cmd (result code): add bean 8 0 93 (NOT_STORED)
3.  [INFO] ++++ data not stored in cache for key: bean
4.  [INFO] ++++ deserializing class com.alisoft.sme.memcached.TBean
5.  name=E网打进
6.  [INFO] ++++ deserializing class com.alisoft.sme.memcached.TBean
7.  name=E网打进
```

声明：ITeye 文章版权属于作者，受法律保护。没有作者书面许可不得转载。

推荐链接

[上海自考专/本科](#)  
12个月,上海自考专/本科签约班 周期短,合格率高,文凭硬,学费低  
[www.zili.cn](#)

Google 提供的广告

- 
- PHP培训费暴跌! 传智播客PHP培训仅3800元!
- 3G培训就业月薪平均7K+, 不3K就业不花一分钱!

返回顶楼

主页

资料

短信

留言

关注

by5739  
等级: 初级会员

发表时间: 2009-06-29

为什么第二次打印出来的不是"E网打进\_修改的"



文章: 42  
积分: 40  
来自: ...



返回顶楼

主页

资料

短信

留言

关注

回帖地址

0

0

请登录投票

oxromantic  
等级: 初级会员

发表时间: 2009-06-30

因为没用cache.replace("bean", tb);



文章: 27

积分: 30  
来自: ...  

我现在离线

返回顶楼

主页

资料

短信

留言

关注

回帖地址

✓

2

✗

0

请登录投票

[论坛首页](#) → [Java编程和Java企业应用版](#)

跳转论坛: 

JavaJavaJava

- [北京: 美团诚聘研发工程师 \(Java\)](#)
- [上海: brite:bill诚聘中级Java程序员](#)
- [湖北: 广通信达杭州研发中心诚聘java工程师](#)
- [上海: 为为网诚聘JAVA软件开发工程师:](#)
- [北京: 胡莱游戏诚聘年薪8-16万高级Java程序员](#)
- [北京: IT之家招聘和猎头诚聘IT公司网站诚聘Java](#)