网上关于Memcached的文章很多，但据我观察，大多是互相转载或者抄袭的，千篇一律。有些则是直接整理的一些超链接然后贴出来。那些超链接笔者大概都进去看了，其实关于Memcached的中文的技术文章，也就那么几篇，优秀的也就更少了。也许是笔者理解、学习能力太差了吧。最近在.NET项目中可能要使用到Memcached这款出色的分布式缓存系统，所以笔者就笨鸟先飞，先研究了一下下。下面我会将总结一下自己研究的结果，并用实例向大家讲解。（入门级别的，高手请直接跳过）。

关于：Memcached的简介，Memcached的实现原理，Memcached的优势.......等等不是本文的重点。所以直接跳过。

（我已经将本文中必要的源码和类库等上传到了csdn下载频道。避免很多英语不好的朋友，访问国外站点下载时候遇到一些困难，浪费时间）。

由于最近的项目是使用.NET作为开发平台，而且使用的是Windows server 2008作服务器。所以，首先要找到Windows平台下的Memcached版本。

下面，我将下载地址贴出来。

<http://download.csdn.net/detail/dinglang_2009/3733784>      或者            [点击此处直接下载](http://download.csdn.net/detail/dinglang_2009/3733784)

下载好了Memcached之后，解压到任意目录下，例如：

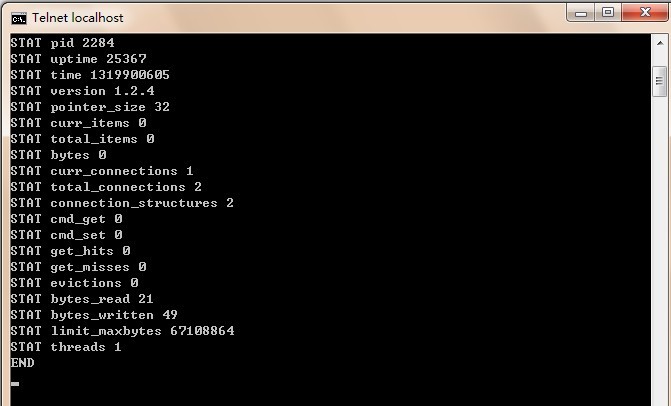
 1. 解压缩文件到c:/memcached      进入cmd控制台（该不会有人问我这个cmd要怎么进去吧？）  
    2. 命令行输入 'c:/memcached/memcached.exe -d install'       --安装 Memcached  
    3. 命令行输入 'c:/memcached/memcached.exe -d start' ，该命令启动 Memcached ，默认监听端口为 11211。

其实可以通过        memcached.exe -p 11121 -m 64    指定它的监听端口是11121（这是它的默认端口，可以指定为其它大于1024的端口，因为小于1024的端口已经有了默认指定），最大使用内存为64m，如果启用了Windows防火墙，切记要在防火墙上打开这个端口。

注:通过 memcached.exe -h 可以查看其帮助     这里不多做介绍了

启动成功之后，在客户端还可以通过telnet来查看和操作Memcached，前提是服务器端和客户端都支持Telnet协议，在Windows7和Windows2008中基于安全性的考虑，默认都不支持，需要在控制面板中安装和启用。（具体的操作我就不说了）

telnet localhost 11121  
连接之后会出现一个命令行窗口，在这个命令行窗口中输入"stats"就可以看到当前Memcached的状态，如下图所示：



启动并配置好Memcached的服务端之后呢？我们下面该准备基于.NET平台的客户端了：

据我总结，大概常见的有三四种吧。必要的类库或者源码，我都将提供给大家下载。

1）.NET memcached client library

Memcached .NET客户端的类库，目前大概只支持.NET1.0和.NET2.0，笔者没仔细研究这个，估计这个已经过时啦。

2）enyim.com Memcached Client

源码地址：[点击下载源码](http://download.csdn.net/detail/dinglang_2009/3733802)        （开源项目，可提供研究学习）

3）Memcached Providers

下载地址：[点击下载](http://download.csdn.net/detail/dinglang_2009/3733770)

Memcached Providers的官网上有一份PDF的文档，是教你怎么配置的。英语不好的童鞋，可以直接参考我的配置（稍后会讲解）。

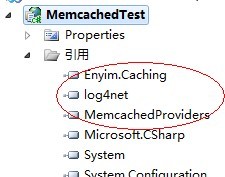
4) BeIT Memcached

这个笔者没用过，不做讲解了

下载完成，解压后会发现，enyim.com Memcached Client中，还包含了著名的Log4net日志框架。而Memcached Providers中其实包含了Enyim.Caching.dll，也包含了Log4netDLL。Memcached Providers更强大，是对enyim.com Memcached Client的又一次封装和扩展。当然，使用和配置起来也同样简单，方便，只是对于一些负责的操作和处理方面，功能更强大一些。

如果只是简单的应用，推荐直接使用enyim.com Memcached Client。除了添加必要的Enyim.Caching.dll引用，还需要修改应用程序的配置文件。

下面我将贴出这两种方式的配置文件和测试代码：（放心，有详细的注释）



测试代码（实现最简单的功能）

[view plain](http://blog.csdn.net/dinglang_2009/article/details/6917794)

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.Linq;
4. **using** System.Web;
5. **using** System.Web.UI;
6. **using** System.Web.UI.WebControls;
7. **using** Enyim.Caching;
8. **using** MemcachedProviders.Cache;
9. **using** System.Threading;
11. **namespace** MemcachedProject
12. {
13. **public** partial **class** \_Default : System.Web.UI.Page
14. {
15. **protected** **void** Page\_Load(**object** sender, EventArgs e)
16. {
17. testMemcachedProviders();
18. }
20. /// <summary>
21. /// 测试Enyim.Caching（注意：Enyim.Caching只具有MemcachedProviders的一部分功能，后者更强大。还可以集成Log4Net日志框架）
22. /// </summary>
23. **public** **void** testEnyimCaching()
24. {
25. MemcachedClient client = **new** MemcachedClient("enyim.com/memcached");
26. //存值  --不带过期时间的存储，Memcached将根据LRU来决定过期策略
27. **bool** result = client.Store(Enyim.Caching.Memcached.StoreMode.Add, "name", "dinglang");
28. //带过期时间的缓存
29. //bool success = client.Store(StoreMode.Add, person.UserName, person, DateTime.Now.AddMinutes(10));
30. **if** (result)
31. {
32. Response.Write("成功存入缓存");
34. //取值
35. **object** name = client.Get("name");
36. **if** (name != **null**)
37. {
38. Response.Write("取出的值为:"+name);
39. }
40. **else**
41. {
42. Response.Write("取值失败");
43. }
44. }
45. **else**
46. {
47. Response.Write("存入缓存失败");
48. }
49. }
50. /// <summary>
51. /// 使用MemcachedProviders客户端
52. /// </summary>
53. **public** **void** testMemcachedProviders()
54. {
55. **string** key = "myName";
56. **string** value = "Dylan";
57. **bool** result =**false**;
58. **string** val=**string**.Empty;
59. #region 存/取最简单的数据类型
60. //如果缓存中没有，就尝试着去存入缓存
61. **if** (DistCache.Get(key) == **null**)
62. {
63. //DistCache.DefaultExpireTime = 1200;//缓存时间
64. result = DistCache.Add(key, value);           //存数据
65. **if** (result)
66. {
67. //如果存入成功，就试着去取
68. Thread.Sleep(500);
69. **string** ret = (**string**)DistCache.Get(key);            //读数据
70. //Assert.AreEqual(value, ret);                     //验证
71. **if** (ret != **null**)
72. {
73. Response.Write(ret);
74. Response.Write("<br/>");
75. }
76. **else**
77. {
78. //取出来的值为null，直接移除该缓存对象
79. DistCache.Remove(key);//移除
80. // DistCache.RemoveAll();//移除所有
81. }
82. }
83. }
84. **else** {
85. //缓存中有，直接拿数据
86. **string** ret = (**string**)DistCache.Get(key);
87. **if** (ret != **null**)
88. {
89. Response.Write(ret);
90. Response.Write("<br/>");
91. }
92. **else** {
93. DistCache.Remove(key);
94. }
95. }
96. #endregion
98. #region  存/取一个Person对象
99. Person person = **new** Person() {Id=007,Name="Dylan"};//new 一个Person对象的实例
100. //如果缓存中没有，则尝试着放入缓存
101. **if** (DistCache.Get<Person>("myObj") == **null**)
102. {
103. result = DistCache.Add("myObj", person);
104. **if** (result)
105. {
106. Thread.Sleep(500);
107. val  = DistCache.Get("myObj").ToString();
108. **if** (val != **null**) {
109. Response.Write(val);
110. Response.Write("<br/>");
111. }
112. **else** {
113. DistCache.Remove("myObj");
114. }
116. }
117. }
118. **else**
119. {
120. //缓存中已经有该对象，就直接从缓存取
121. Person p = DistCache.Get<Person>("myObj");
122. val = person.ToString();
123. //也可以直接这样取
124. // val = DistCache.Get("myObj").ToString();
126. **if** (val != **null**)
127. {
128. Response.Write(val);
129. Response.Write("<br/>");
130. }
131. **else**
132. {
133. DistCache.Remove("myObj");
134. }
136. }
137. #endregion
138. }
139. }
140. }

Person类的定义代码：

[view plain](http://blog.csdn.net/dinglang_2009/article/details/6917794)

1. [Serializable]
2. **public** **class** Person
3. {
4. **private** **int** id;
6. **public** **int** Id
7. {
8. **get** { **return** id; }
9. **set** { id = value; }
10. }
11. **private** **string** name;
13. **public** **string** Name
14. {
15. **get** { **return** name; }
16. **set** { name = value; }
17. }
19. /// <summary>
20. /// 重写Tostring()，方便输出验证
21. /// </summary>
22. /// <returns></returns>
23. **public** **override** **string** ToString()
24. {
25. **return** "Person:" + "{name:" + Name + ",id:" + Id + "}";
26. }
27. }

配置信息：

[view plain](http://blog.csdn.net/dinglang_2009/article/details/6917794)

1. **<?xml** version="1.0"**?>**
3. <!--
4. 有关如何配置 ASP.NET 应用程序的详细信息，请访问
5. http://go.microsoft.com/fwlink/?LinkId=169433
6. --**>**
8. **<configuration>**
10. <!--Enyim.Caching配置（省略了Log4Net框架）   For Memcached-->
12. <!--**<configSections>**
13. **<sectionGroup** name="enyim.com"**>**
14. **<section** name="memcached" type="Enyim.Caching.Configuration.MemcachedClientSection, Enyim.Caching" **/>**
15. **</sectionGroup>**
16. **</configSections>**
17. **<enyim.com** protocol="Binary"**>**
18. **<memcached>**
19. **<servers>**
20. **<add** address="127.0.0.1" port="11121"  **/>**
21. **</servers>**
22. **<socketPool** minPoolSize="10" maxPoolSize="100" connectionTimeout="00:00:10" deadTimeout="00:02:00" **/>**
23. **</memcached>**
24. **</enyim.com>**--**>**
26. <!--The Enyim.Caching配置 End -->
28. <!--MemcachedProviders的配置  -->
29. **<configSections>**
30. **<section** name="cacheProvider" type="MemcachedProviders.Cache.CacheProviderSection, MemcachedProviders"
31. allowDefinition="MachineToApplication" restartOnExternalChanges="true"**/>**
32. **<sectionGroup** name="enyim.com"**>**
33. **<section** name="memcached" type="Enyim.Caching.Configuration.MemcachedClientSection, Enyim.Caching" **/>**
34. **</sectionGroup>**
35. **<section** name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler,log4net"**/>**
36. **</configSections>**
37. **<cacheProvider** defaultProvider="MemcachedCacheProvider"**>**
38. **<providers>**
39. **<add** name="MemcachedCacheProvider"
40. type="MemcachedProviders.Cache.MemcachedCacheProvider, MemcachedProviders"  keySuffix="\_MySuffix\_" defaultExpireTime="2000"**/>**
41. **</providers>**
42. **</cacheProvider>**
43. **<enyim.com>**
44. **<memcached>**
45. **<servers>**
46. <!-- put your own server(s) here-->
47. **<add** address="127.0.0.1" port="11121" **/>**
48. **</servers>**
49. **<socketPool** minPoolSize="10" maxPoolSize="100" connectionTimeout="00:00:10" deadTimeout="00:02:00" **/>**
50. **</memcached>**
51. **</enyim.com>**
53. **<log4net>**
54. **<appender** name="ConsoleAppender" type="log4net.Appender.ConsoleAppender"**>**
55. **<layout** type="log4net.Layout.PatternLayout"**>**
56. **<conversionPattern** value="%date [%thread] %-5level %logger [%property{NDC}]- %message%newline" **/>**
57. **</layout>**
58. **</appender>**
59. **<root>**
60. **<priority** value="WARN"**/>**
61. **<appender-ref** ref="ConsoleAppender"**>**
62. **<filter** type="log4net.Filter.LevelRangeFilter"**>**
63. **<levelMin** value="WARN"**/>**
64. **<levelMax** value="FATAL"**/>**
65. **</filter>**
66. **</appender-ref>**
67. **</root>**
68. **</log4net>**
70. <!--The MemcachedProviders配置 End-->
72. **<connectionStrings>**
73. **<add** name="ApplicationServices"
74. connectionString="data source=.\SQLEXPRESS;Integrated Security=SSPI;AttachDBFilename=|DataDirectory|\aspnetdb.mdf;User Instance=true"
75. providerName="System.Data.SqlClient" **/>**
76. **</connectionStrings>**

79. **<system.web>**
80. **<compilation** debug="true" targetFramework="4.0" **/>**
82. **<authentication** mode="Forms"**>**
83. **<forms** loginUrl="~/Account/Login.aspx" timeout="2880" **/>**
84. **</authentication>**
86. **<membership>**
87. **<providers>**
88. **<clear/>**
89. **<add** name="AspNetSqlMembershipProvider" type="System.Web.Security.SqlMembershipProvider" connectionStringName="ApplicationServices"
90. enablePasswordRetrieval="false" enablePasswordReset="true" requiresQuestionAndAnswer="false" requiresUniqueEmail="false"
91. maxInvalidPasswordAttempts="5" minRequiredPasswordLength="6" minRequiredNonalphanumericCharacters="0" passwordAttemptWindow="10"
92. applicationName="/" **/>**
93. **</providers>**
94. **</membership>**
96. **<profile>**
97. **<providers>**
98. **<clear/>**
99. **<add** name="AspNetSqlProfileProvider" type="System.Web.Profile.SqlProfileProvider" connectionStringName="ApplicationServices" applicationName="/"**/>**
100. **</providers>**
101. **</profile>**
103. **<roleManager** enabled="false"**>**
104. **<providers>**
105. **<clear/>**
106. **<add** name="AspNetSqlRoleProvider" type="System.Web.Security.SqlRoleProvider" connectionStringName="ApplicationServices" applicationName="/" **/>**
107. **<add** name="AspNetWindowsTokenRoleProvider" type="System.Web.Security.WindowsTokenRoleProvider" applicationName="/" **/>**
108. **</providers>**
109. **</roleManager>**
111. **</system.web>**



116. **<system.webServer>**
117. **<modules** runAllManagedModulesForAllRequests="true"**/>**
118. **</system.webServer>**
119. **</configuration>**

好的，关于Memcached的基础部分，就先讲到这里。后续，我将会整理一些关于Memcached的高级技术，包括：复杂的存储(缓存/更新)控制模型，缓存策略，在分布式、集群、负载均衡的环境下的使用和管理技巧，性能监控，日志记录等.敬请期待。