**一、引言**

        今天开始有关Redis学习的第九篇文章了，以后肯定会大量系统使用Redis作为缓存介质，为了更好的更好的Redis，自己写了两个工具类，但是这两个工具类，没有提供一致的接口，是为了使用的独立性。测试已经完毕，可以正常访问Windows和Linux版本上的Redis服务，各种操作也没问题。今天就把主要代码贴出来，因为有一部分是需要配置文件的，我自己独立写了一套配置系统，这套配置系统就不贴出来，大家可以根据自己的理解来写这个部分的内容，好了，开始我们今天的写作吧。

**二、简介**

        我写了两个工具类，第一个是以【StackExchange.Redis】为实现技术的Redis的客户端工具类，另一个是以【ServiceStack.Redis】为实现技术的Redis客户端工具类。【ServiceStack.Redis】的官网主页：[https://servicestack.net/redis](https://servicestack.net/redis" \t "http://www.cnblogs.com/PatrickLiu/p/_blank)，【ServiceStack.Redis】在github上的主页：[https://github.com/ServiceStack/ServiceStack.Redis](https://github.com/ServiceStack/ServiceStack.Redis" \t "http://www.cnblogs.com/PatrickLiu/p/_blank)，【ServiceStack.Redis】和Visual Studio 2015整个也很方便，微软在这块做的的很不过，我们可以通过命令直接安装：Install-Package ServiceStack.Redis，当然我们也可以通过NuGet来安装，安装完成，在您项目的【引用】里面会生成四个dll，分别是：ServiceStack.Common，ServiceStack.Interfaces，ServiceStack.Redis，ServiceStack.Client 四个程序集，然后在项目中使用Using引入就可以正常使用了。【ServiceStack.Redis】是Redis官方推荐的C#客户端，性能非常优越，使用也很方便，但是从v4版本已经逐渐商业化了，目的可以想而知，嗨，都是钱惹的祸。

       上面我说道了，【ServiceStack.Redis】后来就商业化了，我想了想光靠他也是不行了，还要在写一个版本的工具类，所以就选择了【StackExchange.Redis】，【StackExchange.Redis】也是开源的，到目前，一直都是开源的，它的地址是：[https://github.com/StackExchange/StackExchange.Redis](https://github.com/StackExchange/StackExchange.Redis" \t "http://www.cnblogs.com/PatrickLiu/p/_blank)，我使用的是.net 4.5，工具采用 vs2015， StackExchange.Redis版本是1.0.488。工具类还会在持续的使用过程中，我还会更新和修改的。

**三、实例代码**   
         这是Redis客户端的配置文件的格式，格式很简单，可以分开配置，也可以合在一起配置。**代码中标红的是和我的配置系统有关的代码，大家请注意。**

[IMG_256](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

<configuration>

<configSections>

<section name="Framework" type="Enterprise.Framework.Configuration.ConfigurationFrameworkSectionHandler,Enterprise.Framework.Configuration" />

</configSections>

<Framework type="Enterprise.Framework.Configuration.ConfigurationFrameworkManager,Enterprise.Framework.Configuration">

<Enterprise.Framework.NoSQL>

<RedisClients>

<RedisClient name="ServiceStack" technology="ServiceStack.Redis">

<readWriteHosts>192.168.3.11:6379,192.168.3.11:6380</readWriteHosts>

<!--<readOnlyHosts>可以省略该项</readOnlyHosts>-->

<!--<maxWritePoolSize>可以省略该项</maxWritePoolSize>-->

<!--<maxReadPoolSize>可以省略该项</maxReadPoolSize>-->

<!--<password>可以省略该项</password>-->

<!--<autoStart>可以省略该项</autoStart>-->

</RedisClient>

<RedisClient name="StackExchange" technology="StackExchange.Redis">

<host>192.168.131.1:6379</host>

<!--<password>可以省略该项</password>-->

</RedisClient>

</RedisClients>

</Enterprise.Framework.NoSQL>

</Framework>

</configuration>

[IMG_257](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**1、这是以【ServiceStack.Redis】为实现技术的工具类，对外界的访问接口提供了2个，第一个是以配置文件中自定义的名称参数的，红色代码是和我独立的配置系统相关联的，另一个访问接口是以配置实体类参数的，代码很简单，不多说了。**

**工具类：ServiceStackRedisClientProvider.cs**

[IMG_258](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

1 /// <summary> 2 /// 通过ServiceStack.Redis实现的Redis的客户端操作类型 3 /// </summary> 4 public sealed class ServiceStackRedisClientProvider 5 { 6 #region 私有变量 7 8 //线程同步变量 9 private static readonly object lockObject = new object(); 10 11 //redis链接池管理对象 12 private static volatile PooledRedisClientManager \_instance = null; 13 14 //配置文件里面的ServiceStack详细配置设置 15 private static ServiceStackDetails \_serviceStackDetails; 16 17 //可以自行配置ServiceStack的配置对象 18 private static ServiceStackConfigEntry \_serviceStackConfigEntry; 19 20 #endregion 21 22 #region 私有构造函数 23 24 /// <summary> 25 /// 私有构造函数，禁止外部通过new关键字来创建该对象实例 26 /// </summary> 27 private ServiceStackRedisClientProvider() { } 28 29 #endregion 30 31 #region 获取PooledRedisClientManager实例的方法 32 33 /// <summary> 34 /// 获取redis链接池管理对象实例 35 /// 实例发生变化的集中情况： 36 /// 1.实例为空 37 /// 2.配置文件发生变化 38 /// </summary> 39 /// <param name="startByConfigFile">这是一个布尔值，true表示根据配置文件的配置启动，false表示是根据配置对象启动</param> 40 /// <returns>返回PooledRedisClientManager类型的对象实例</returns> 41 private static PooledRedisClientManager GetInstance(bool startByConfigFile) 42 { 43 if (\_instance == null) 44 { 45 lock (lockObject) 46 { 47 if (\_instance == null) 48 { 49 string[] readWriteServerList=null; 50 string[] readOnlyServerList=null; 51 RedisClientManagerConfig managerConfig=null; 52 53 //根据我们配置文件中数据来设置启动信息（app.config或者web.config） 54 if (startByConfigFile && (\_serviceStackDetails != null)) 55 { 56 managerConfig = new RedisClientManagerConfig() 57 { 58 AutoStart = \_serviceStackDetails.AutoStart, 59 MaxReadPoolSize = \_serviceStackDetails.MaxReadPoolSize, 60 MaxWritePoolSize = \_serviceStackDetails.MaxWritePoolSize, 61 }; 62 63 readWriteServerList = GetRedisHosts(\_serviceStackDetails.ReadWriteHosts); 64 readOnlyServerList = GetRedisHosts(\_serviceStackDetails.ReadOnlyHosts); 65 } 66 else if (!startByConfigFile && (\_serviceStackConfigEntry != null))//根据配置对象来设置启动信息(ServiceStackConfigEntry) 67 { 68 managerConfig = new RedisClientManagerConfig() 69 { 70 AutoStart = \_serviceStackConfigEntry.AutoStart, 71 MaxReadPoolSize = \_serviceStackConfigEntry.MaxReadPoolSize, 72 MaxWritePoolSize = \_serviceStackConfigEntry.MaxWritePoolSize, 73 }; 74 75 readWriteServerList = GetRedisHosts(\_serviceStackConfigEntry.ReadWriteHosts); 76 readOnlyServerList = GetRedisHosts(\_serviceStackConfigEntry.ReadOnlyHosts); 77 } 78 else 79 { 80 throw new InvalidOperationException("Redis客户端初始化配置失败！"); 81 } 82 83 if ((readWriteServerList != null && readWriteServerList.Length > 0)&&(readOnlyServerList != null && readOnlyServerList.Length <= 0)) 84 { 85 \_instance = new PooledRedisClientManager(readWriteServerList); 86 } 87 88 if ((readWriteServerList != null && readWriteServerList.Length > 0) && (readOnlyServerList != null && readOnlyServerList.Length > 0)) 89 { 90 \_instance = new PooledRedisClientManager(readWriteServerList, readOnlyServerList, managerConfig); 91 } 92 } 93 } 94 } 95 return \_instance; 96 } 97 98 /// <summary> 99 /// 解析Redis服务器列表，该列表格式IP[:Port]100 /// </summary>101 /// <param name="redisHosts">包含一个或者多个Redis服务器地址的字符串列表，以逗号做为分隔符</param>102 /// <returns>返回Redis服务器地址列表</returns>103 private static string[] GetRedisHosts(string redisHosts)104 {105 if (string.IsNullOrWhiteSpace(redisHosts) || string.IsNullOrEmpty(redisHosts))106 {107 return new string[] { };108 }109 var hosts=redisHosts.Split(',');110 foreach (var host in hosts)111 {112 if (!Regex.IsMatch(host, @"^(25[0-5]|2[0-4][0-9]|[0-1]{1}[0-9]{2}|[1-9]{1}[0-9]{1}|[1-9])\.(25[0-5]|2[0-4][0-9]|[0-1]{1}[0-9]{2}|[1-9]{1}[0-9]{1}|[1-9]|0)\.(25[0-5]|2[0-4][0-9]|[0-1]{1}[0-9]{2}|[1-9]{1}[0-9]{1}|[1-9]|0)\.(25[0-5]|2[0-4][0-9]|[0-1]{1}[0-9]{2}|[1-9]{1}[0-9]{1}|[0-9]):\d{3,4}$"))113 {114 throw new InvalidOperationException("Redis服务器地址格式不正确！");115 }116 }117 return hosts;118 }119 120 #endregion121 122 #region 提供对外访问接口123 124 /// <summary>125 /// 获取Redis客户端对象实例126 /// </summary>127 /// <param name="redisClientName">在配置文件中，Redis客户端的名称</param>128 /// <param name="databaseIndex">redis逻辑分为16个数据库，排序为：0-15，我们默认使用的是0号数据库，数据库当前的索引值</param>129 /// <returns>返回IRedisClient对象实例</returns>130 public static IRedisClient GetRedisClient(string redisClientName,int databaseIndex = 0)131 {132 //获取配置数据133 ParameterValidityChecker.RequiredParameterStringNotNullOrWhiteSpace(redisClientName, "Redis客户端的名称不能为空！");134 var \_configurationManager = (ConfigurationFrameworkManager)ConfigurationManager.GetSection("Framework");135 if (\_configurationManager != null)136 {137 \_serviceStackDetails = \_configurationManager.RedisClientConfiguration.GetServiceStackDetails(redisClientName);138 if (\_serviceStackDetails == null)139 {140 throw new InvalidOperationException("以ServiceStack.Redis为实现技术的Redis客户端的配置有误！");141 }142 }143 else144 {145 throw new InvalidOperationException("以ServiceStack.Redis为实现技术的Redis客户端的配置有误！");146 }147 148 //实例化Redis客户端实例对象149 var pooledRedisClientManager = GetInstance(true);150 var redisClient = pooledRedisClientManager.GetClient();151 if (!string.IsNullOrEmpty(\_serviceStackDetails.Password))152 {153 redisClient.Password = \_serviceStackDetails.Password;154 }155 redisClient.Db = databaseIndex;156 return redisClient;157 }158 159 /// <summary>160 /// 获取Redis客户端对象实例161 /// </summary>162 /// <param name="serviceStackConfigEntry">在配置文件中，Redis客户端的名称</param>163 /// <param name="databaseIndex">redis逻辑分为16个数据库，排序为：0-15，我们默认使用的是0号数据库，数据库当前的索引值</param>164 /// <returns>返回IRedisClient对象实例</returns>165 public static IRedisClient GetRedisClient(ServiceStackConfigEntry serviceStackConfigEntry, int databaseIndex = 0)166 {167 //获取配置数据168 if (serviceStackConfigEntry == null)169 {170 throw new ArgumentNullException("以ServiceStack.Redis为实现技术的Redis客户端的配置对象不能为空！");171 }172 else173 {174 \_serviceStackConfigEntry = serviceStackConfigEntry;175 }176 177 if (string.IsNullOrEmpty(\_serviceStackConfigEntry.ReadWriteHosts) || string.IsNullOrWhiteSpace(\_serviceStackConfigEntry.ReadWriteHosts))178 {179 throw new InvalidOperationException("【ReadWriteHosts】必须设置其值！");180 }181 182 //实例化Redis客户端实例对象183 var pooledRedisClientManager = GetInstance(false);184 var redisClient = pooledRedisClientManager.GetClient();185 if (!string.IsNullOrEmpty(\_serviceStackConfigEntry.Password)&&!string.IsNullOrWhiteSpace(\_serviceStackConfigEntry.Password))186 {187 redisClient.Password = \_serviceStackConfigEntry.Password;188 }189 redisClient.Db = databaseIndex;190 return redisClient;191 }192 193 #endregion194 }

[IMG_259](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**配置实体类 ServiceStackConfigEntry.cs的代码：**

[IMG_260](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

1 /// <summary> 2 /// 配置文件中，以ServiceStack.Redis为实现技术的配置Redis的详情 3 /// </summary> 4 public sealed class ServiceStackConfigEntry 5 { 6 #region 构造函数 7 8 /// <summary> 9 /// 给配置参数初始化默认值10 /// </summary>11 public ServiceStackConfigEntry()12 {13 ReadWriteHosts = "127.0.0.1:6379";14 ReadOnlyHosts = string.Empty;15 MaxWritePoolSize = 200;16 MaxReadPoolSize = 200;17 Password = string.Empty;18 AutoStart = true;19 }20 21 #endregion22 23 #region 配置属性24 25 /// <summary>26 /// 可读可写的Redis服务器地址，多个地址以逗号分隔，例如：192.168.127.11:6379,192.168.127.128:638027 /// </summary>28 public string ReadWriteHosts { get; set; }29 30 /// <summary>31 /// 只能读的Redis服务器地址，多个地址以逗号分隔，例如：192.168.127.11:6379,192.168.127.128:638032 /// </summary>33 public string ReadOnlyHosts { get; set; }34 35 /// <summary>36 /// 最大写链接数37 /// </summary>38 public int MaxWritePoolSize { get; set; }39 40 /// <summary>41 /// 最大读链接数42 /// </summary>43 public int MaxReadPoolSize { get; set; }44 45 /// <summary>46 /// 登陆Redis服务器的密码47 /// </summary>48 public string Password { get; set; }49 50 /// <summary>51 /// 是否自动启动52 /// </summary>53 public bool AutoStart { get; set; }54 55 #endregion56 }

[IMG_261](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**获取配置文件详情的类型：ServiceStackDetails.cs**

[IMG_262](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

/// <summary>

/// 配置文件中，以ServiceStack.Redis为实现技术的配置Redis的详情

/// </summary>

public sealed class ServiceStackDetails

{

#region 构造函数

/// <summary>

/// 给配置参数初始化默认值

/// </summary>

public ServiceStackDetails()

{

ReadWriteHosts = "127.0.0.1:6379";

ReadOnlyHosts = string.Empty;

MaxWritePoolSize = 200;

MaxReadPoolSize = 200;

Password = string.Empty;

AutoStart = true;

}

#endregion

#region 配置属性

/// <summary>

/// 可读可写的Redis服务器地址，多个地址以逗号分隔，例如：192.168.127.11:6379,192.168.127.128:6380

/// </summary>

public string ReadWriteHosts { get; internal set; }

/// <summary>

/// 只能读的Redis服务器地址，多个地址以逗号分隔，例如：192.168.127.11:6379,192.168.127.128:6380

/// </summary>

public string ReadOnlyHosts { get; internal set; }

/// <summary>

/// 最大写链接数

/// </summary>

public int MaxWritePoolSize { get; internal set; }

/// <summary>

/// 最大读链接数

/// </summary>

public int MaxReadPoolSize { get; internal set; }

/// <summary>

/// 登陆Redis服务器的密码

/// </summary>

public string Password { get; internal set; }

/// <summary>

/// 是否自动启动

/// </summary>

public bool AutoStart { get; internal set; }

#endregion

}

[IMG_263](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**2、这是以【StackExchange.Redis】为实现技术的工具类，对外界的访问接口提供了2个，第一个是以配置文件中自定义的名称参数的，红色代码是和我独立的配置系统相关联的，另一个访问接口是以配置实体类参数的，代码很简单，不多说了。**

**工具类： StackExchangeRedisClientProvider.cs**

[IMG_264](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

1 /// <summary> 2 /// 通过StackExchange.Redis实现的Redis的客户端操作类型 3 /// </summary> 4 public sealed class StackExchangeRedisClientProvider 5 { 6 #region 私有字段 7 8 /// <summary> 9 /// 线程同步变量 10 /// </summary> 11 private static readonly object lockObject = new object(); 12 13 /// <summary> 14 /// redis链接池管理对象 15 /// </summary> 16 private static volatile ConnectionMultiplexer \_instance; 17 18 /// <summary> 19 /// 日志记录器 20 /// </summary> 21 private static readonly ILog \_log = LogManager.GetLogger(typeof(StackExchangeRedisClientProvider)); 22 23 private static StackExchangeDetails \_stackExchangeDetails; 24 25 private static StackExchangeConfigEntry \_stackExchangeConfigEntry; 26 27 #endregion 28 29 #region 私有构造函数 30 31 /// <summary> 32 /// 私有构造函数，禁止不允许通过new 来实例化该对象 33 /// </summary> 34 private StackExchangeRedisClientProvider() { } 35 36 #endregion 37 38 #region 获取Redis客户端实例 39 40 /// <summary> 41 /// 使用一个静态属性来返回已连接的实例 42 /// 实例发生变化的几种情况： 43 /// 1.实例为空 44 /// 2.连接关闭 45 /// 3.文件发生变化时 46 /// </summary> 47 /// <param name="startByConfigFile">这是一个布尔值，true表示根据配置文件的配置启动，false表示是根据配置对象启动</param> 48 /// <returns>返回ConnectionMultiplexer类型的对象实例</returns> 49 private static ConnectionMultiplexer GetInstance(bool startByConfigFile) 50 { 51 if (startByConfigFile) 52 { 53 GetRedisHosts(\_stackExchangeDetails.Hosts); 54 } 55 else 56 { 57 GetRedisHosts(\_stackExchangeConfigEntry.Hosts); 58 } 59 60 if (\_instance == null || !\_instance.IsConnected) 61 { 62 lock (lockObject) 63 { 64 if (\_instance == null || !\_instance.IsConnected) 65 { 66 if (startByConfigFile) 67 { 68 \_instance = ConnectionMultiplexer.Connect(\_stackExchangeDetails.Hosts); 69 } 70 else 71 { 72 \_instance = ConnectionMultiplexer.Connect(\_stackExchangeConfigEntry.Hosts); 73 } 74 } 75 } 76 } 77 \_instance.ErrorMessage += MuxerErrorMessage; 78 \_instance.HashSlotMoved += MuxerHashSlotMoved; 79 \_instance.InternalError += MuxerInternalError; 80 \_instance.ConnectionFailed += MuxerConnectionFailed; 81 \_instance.ConnectionRestored += MuxerConnectionRestored; 82 \_instance.ConfigurationChanged += MuxerConfigurationChanged; 83 return \_instance; 84 } 85 86 /// <summary> 87 /// 解析Redis服务器列表，该列表格式IP:Port 88 /// </summary> 89 /// <param name="redisHosts">包含一个或者多个Redis服务器地址的字符串列表，以逗号做为分隔符</param> 90 private static void GetRedisHosts(string redisHosts) 91 { 92 if (string.IsNullOrWhiteSpace(redisHosts) || string.IsNullOrEmpty(redisHosts)) 93 { 94 return; 95 } 96 var hosts = redisHosts.Split(','); 97 foreach (var host in hosts) 98 { 99 if (!Regex.IsMatch(host, @"^(25[0-5]|2[0-4][0-9]|[0-1]{1}[0-9]{2}|[1-9]{1}[0-9]{1}|[1-9])\.(25[0-5]|2[0-4][0-9]|[0-1]{1}[0-9]{2}|[1-9]{1}[0-9]{1}|[1-9]|0)\.(25[0-5]|2[0-4][0-9]|[0-1]{1}[0-9]{2}|[1-9]{1}[0-9]{1}|[1-9]|0)\.(25[0-5]|2[0-4][0-9]|[0-1]{1}[0-9]{2}|[1-9]{1}[0-9]{1}|[0-9]):\d{3,4}$"))100 {101 throw new InvalidOperationException("Redis服务器地址格式不正确！");102 }103 }104 }105 106 /// <summary>107 /// 获取Redis客户端对象实例108 /// </summary>109 /// <param name="redisClientName">在配置文件中，Redis客户端的名称</param>110 /// <param name="databaseIndex">redis逻辑分为16个数据库，排序为：0-15，我们默认使用的是0号数据库，数据库当前的索引值</param>111 /// <returns></returns>112 public static IDatabase GetRedisClient(string redisClientName, int databaseIndex =0)113 {114 //获取配置数据115 ParameterValidityChecker.RequiredParameterStringNotNullOrWhiteSpace(redisClientName, "Redis客户端的名称不能为空！");116 var \_configurationManager = (ConfigurationFrameworkManager)ConfigurationManager.GetSection("Framework");117 if (\_configurationManager != null)118 {119 \_stackExchangeDetails = \_configurationManager.RedisClientConfiguration.GetStackExchangeDetails(redisClientName);120 if (\_stackExchangeDetails == null)121 {122 throw new InvalidOperationException("以StackExchange.Redis为实现技术的Redis客户端的配置有误！");123 }124 }125 else126 {127 throw new InvalidOperationException("以StackExchange.Redis为实现技术的Redis客户端的配置有误！");128 }129 130 //实例化Redis客户端实例对象131 var instance = GetInstance(true);132 return instance.GetDatabase(databaseIndex);133 }134 135 /// <summary>136 /// 获取Redis客户端对象实例137 /// </summary>138 /// <param name="stackExchangeConfigEntry">StackExchange配置对象</param>139 /// <param name="databaseIndex">redis逻辑分为16个数据库，排序为：0-15，我们默认使用的是0号数据库，数据库当前的索引值</param>140 /// <returns></returns>141 public static IDatabase GetRedisClient(StackExchangeConfigEntry stackExchangeConfigEntry, int databaseIndex =0)142 {143 //获取配置数据144 if (stackExchangeConfigEntry == null)145 {146 throw new ArgumentNullException("以StackExchange.Redis为实现技术的Redis客户端的配置对象不能为空！");147 }148 else149 {150 \_stackExchangeConfigEntry = stackExchangeConfigEntry;151 }152 153 if (string.IsNullOrEmpty(\_stackExchangeConfigEntry.Hosts) || string.IsNullOrWhiteSpace(\_stackExchangeConfigEntry.Hosts))154 {155 throw new InvalidOperationException("【Hosts】必须设置其值！");156 }157 158 //实例化Redis客户端实例对象159 var instance = GetInstance(false);160 return instance.GetDatabase(databaseIndex);161 }162 163 #endregion164 }

[IMG_265](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**配置实体类：StackExchangeConfigEntry.cs**

[IMG_266](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

1 /// <summary> 2 /// 配置文件中，以StackExchange.Redis为实现技术的配置Redis的详情 3 /// </summary> 4 public sealed class StackExchangeConfigEntry 5 { 6 #region 构造函数 7 8 /// <summary> 9 /// 给配置参数初始化默认值10 /// </summary>11 public StackExchangeConfigEntry()12 {13 Hosts = "127.0.0.1:6379";14 Password = string.Empty;15 }16 17 #endregion18 19 #region 配置属性20 21 /// <summary>22 /// Redis服务器地址，多个地址以逗号分隔，例如：192.168.127.11:6379,192.168.127.128:638023 /// </summary>24 public string Hosts { get; set; }25 26 /// <summary>27 /// 登陆Redis服务器的密码28 /// </summary>29 public string Password { get; set; }30 31 #endregion32 }

[IMG_267](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**根据配置信息获取数据的类型：StackExchangeDetails.cs**

[IMG_268](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

1 /// <summary> 2 /// 配置文件中，以StackExchange.Redis为实现技术的配置Redis的详情 3 /// </summary> 4 public sealed class StackExchangeDetails 5 { 6 #region 构造函数 7 8 /// <summary> 9 /// 给配置参数初始化默认值10 /// </summary>11 public StackExchangeDetails()12 {13 Hosts = "127.0.0.1:6379";14 Password = string.Empty;15 }16 17 #endregion18 19 #region 配置属性20 21 /// <summary>22 /// Redis服务器的主机地址，如果多个地址则以逗号分隔，格式：127.0.0.1:6379,127.0.0.1:638023 /// </summary>24 public string Hosts{ get; internal set; }25 26 /// <summary>27 /// 登陆Redis服务器的密码28 /// </summary>29 public string Password { get; internal set; }30 31 #endregion32 }

[IMG_269](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**四、结束**

       好了，今天就写到这里了，先说明一下，这两个类暂时没有提供统一的接口，看以后的需要吧，如果有需要，我在重构。StackExchangeDetails 和 ServiceStackDetails 这两个类在这个 Enterprise.Framework.Configuration 命名空间，配置的系统暂时就不贴代码了，代码很多，其他的类型都在 Enterprise.Framework.NoSQL.RedisClient 这个命名空间下边。