

其实，我还会学狗叫，汪、汪汪！
不玩博客了！

公告

昵称：逆心
园龄：7年7个月
粉丝：2395
关注：31
[+加关注](#)

搜索

找找看

谷歌搜索

常用链接

[我的随笔](#)
[我的评论](#)
[我的参与](#)
[最新评论](#)
[我的标签](#)

随笔分类

[.Net基础\(55\)](#)
[.Net模块：Ado.net \(5\)](#)
[.Net模块：Asp.net\(6\)](#)
[.Net模块：多线程\(7\)](#)
[.Net模块：反射\(3\)](#)
[.Net模块：配置文件\(7\)](#)
[C#集合\(3\)](#)
[MySQL：MySQL基础\(10\)](#)
[MySQL性能\(2\)](#)
[Oracle：Oracle基础\(1\)](#)
[PowerDesigner熟悉\(2\)](#)
[SQLServer：T-SQL语法\(11\)](#)
[SQLServer：查询反模式\(5\)](#)
[SQLServer：存储过程重编译\(2\)](#)
[SQLServer：基础\(18\)](#)
[SQLServer：开发反模式\(3\)](#)
[SQLServer：索引\(13\)](#)
[SQLServer：统计分析\(5\)](#)
[SQLServer：系统函数\(6\)](#)
[SQLServer：性能优化\(8\)](#)
[SQLServer：语句\(8\)](#)
[SQLServer：执行计划\(5\)](#)
[XML：XML学习\(7\)](#)
[测试：LoadRunner\(4\)](#)
[调试：Windbg,SOS\(14\)](#)
[感悟\(1\)](#)
[工具：CodeSmith\(4\)](#)
[工具：Reflector\(1\)](#)
[工具：Visual Studio,SVN\(6\)](#)
[技巧\(7\)](#)
[进程、线程、应用程序域\(2\)](#)
[框架：Asp.net MVC\(14\)](#)

ServiceStack.Redis之IRedisClient<第三篇>

事实上，IRedisClient里面的很多方法，其实就是Redis的命令名。只要对Redis的命令熟悉一点就能够非常快速地理解和掌握这些方法，趁着现在对Redis不是特别了解，我也对着命令来了解一下这些方法。

一、属性

IRedisClient的属性如下：

属性	说明
ConnectTimeout	连接超时
Db	当前数据库的ID或下标
DbSize	当前数据库的 key 的数量
HadExceptions	
Hashes	存储复杂对象，一个value中有几个field
Host	Redis的Server服务器主机地址
Info	返回关于 Redis 服务器的各种信息和统计数值
LastSave	最近一次 Redis 成功将数据保存到磁盘上的时间
Lists	当前数据库中所有的List集合
Password	密码
Port	Redis的Server端口
RetryCount	重试次数
RetryTimeout	重试超时
SendTimeout	发送超时
Sets	当前数据库中所有的HashSet<T>集合
SortedSets	当前数据库中所有的SortedSet<T>集合
this[string key]	通过索引的方式(key)访问一个字符串类型值

代码示例：



```
RClient.AddItemToSet("蜀国", "刘备");  
RClient.AddItemToSet("蜀国", "关羽");
```

- 框架：Autofac(3)
- 框架：HtmlAgilityPack
- HTML解析(8)
- 框架：Ibatis.Net学习(7)
- 框架：Linq学习笔记(7)
- 框架：Lucene.net(8)
- 框架：NHibernate(20)
- 框架：WWF(16)
- 垃圾箱(4)
- 面向对象(1)
- 命名空间：System.IO(12)
- 命名空间：System.Net(1)
- 命名空间：System.Web(8)
- 其他：序列化(2)
- 其他：学习方法(2)
- 其他：正则表达式(7)
- 前端：Bootstrap(3)
- 前端：css学习笔记(8)
- 前端：HTML(6)
- 前端：javascript笔记(37)
- 前端：jQuery笔记(28)
- 前端：Json(6)
- 前端：Open Flash Chart(2)
- 三大数据库差异
- 设计模式(26)
- 数据结构(1)
- 数据库设计：SQL反模式笔记(11)
- 算法：算法(4)
- 协议：HTTP(5)
- 性能：缓存(8)
- 原理：Asp.net 原理(9)

随笔档案

- 2018年11月 (1)
- 2017年2月 (1)
- 2017年1月 (1)
- 2016年7月 (5)
- 2016年5月 (3)
- 2016年4月 (24)
- 2015年10月 (1)
- 2015年7月 (1)
- 2015年6月 (1)
- 2015年2月 (1)
- 2015年1月 (1)
- 2014年12月 (8)
- 2014年11月 (5)
- 2014年10月 (1)
- 2014年9月 (15)
- 2014年8月 (7)
- 2014年7月 (1)
- 2014年6月 (2)
- 2014年5月 (17)
- 2014年4月 (9)
- 2014年3月 (9)
- 2014年2月 (10)
- 2014年1月 (7)
- 2013年12月 (5)
- 2013年11月 (24)
- 2013年10月 (26)
- 2013年9月 (26)

```
RCClient.AddItemToSet("蜀国", "张飞");

IHasNamed<IRedisSet> rr = RCClient.Sets;
HashSet<string> HashSetString = rr["蜀国"].GetAll();
foreach (string str in HashSetString)
{
    Response.Write(str);
}
```



二、IRedisClient数据操作

1、ICacheClient接口

IRedisClient实现了接口ICacheClient，其中ICacheClient主要提供的功能如下：

方法	说明
Add	根据传入的key-value添加一条记录，当key已存在返回false
FlushAll	使所有缓存失效（清除Redis所有数据库的所有Key）
Get	根据传入的key获取一条记录的值得
GetAll	根据传入的多个key获取多条记录的值得
Remove	根据传入的key移除一条记录
RemoveAll	根据传入的多个key移除多条记录
Replace	根据传入的key覆盖一条记录的值得，当key不存在不会添加
Set	根据传入的key修改一条记录的值得，当key不存在则添加
SetAll	根据传入的多个key覆盖多条记录
Increment	
Decrement	

特别说明，比如添加的主要方法包括两个重载，一个多了个DateTime类型参数，一个多了TimeSpan类型的参数。这两个都是缓存失效的时间(相当于缓存依赖里的绝对过期时间)。

- DateTime失效点：**到达该时间点**，立即失效；
- TimeSpan失效点：**经过该时间段**，立即失效；

简单示例：

2013年8月 (2)
2013年7月 (10)
2013年6月 (42)
2013年5月 (31)
2013年4月 (32)
2013年3月 (44)
2013年2月 (30)
2013年1月 (50)
2012年12月 (32)
2012年11月 (7)
2012年10月 (2)

内容

积分与排名

积分 - 949008
排名 - 110

最新评论

1. Re:javascript 实现Hash Table(哈希表)
不知道为什么js里连哈希表都没，这个实现本质上还是遍历查找吧。总感觉效率上会有影响

--日不落帝国
2. Re:百度编辑器Ueditor自动换行，添加<p>的问题
你好，我想问下为什么我这一段换行写之后输入内容光标都自动换行？

--wlnancy
3. Re:SQLServer转MYSQL的方法(连数据)
为什么有的表过不来？已经全选了表啊，求大神指教

--李海鹏
4. Re:SQLServer - 约束很详细，感谢

--Bakom
5. Re:C# 排列组合
非常有用！

--Caiger

阅读排行榜

1. jQuery插件之ajaxFileUpload(361672)
2. MySQL 数据备份与还原(288374)
3. SQL 操作结果集 -并集、差集、交集、结果集排序(182636)
4. SQL语句 - 嵌套查询(171152)
5. jqGrid使用记录(142393)

评论排行榜

1. jQuery插件之ajaxFileUpload(35)



```
public ActionResult Index()
{
    RedisClientManagerConfig RedisConfig = new
RedisClientManagerConfig();
    RedisConfig.AutoStart = true;
    RedisConfig.MaxReadPoolSize = 60;
    RedisConfig.MaxWritePoolSize = 60;

    PooledRedisClientManager prcm = new
PooledRedisClientManager(new List<string>() { "127.0.0.1" },
new List<string>() { "127.0.0.1" }, RedisConfig);

    using (IRedisClient RClient = prcm.GetClient())
    {
        RClient.Add("c1", "缓存1");
        RClient.Set("c1", "缓存2");
        RClient.Replace("c1", "缓存3");
        Response.Write(RClient.Get<string>("c1"));
        RClient.Remove("c1");
        Response.Write(RClient.Get<string>("c1") ==
null);
    }

    return Content("");
}
```



2、简单功能

当然，除了实现ICacheClient接口的功能外，对于基本操作，实际上也还有很多功能

方法	说明
AppendToValue	根据Key将参数value追加到原有值的结尾
ContainsKey	判断Key在本数据库内是否已被使用(包括各种类型、内置集合等等)
GetAllKeys	获取所有的Keys集合
DecrementValue	根据指定的Key，将值减1(仅整型有效)
DecrementValueBy	根据指定的Key，将值减去指定值(仅整型有效)
IncrementValue	根据指定的Key，将值加1(仅整型有效)
IncrementValueBy	根据指定的Key，将值加上指定值(仅整型有效)
RenameKey	重命名一个Key，值不变

2. HtmlAgilityPack 之 HtmlNode类(16)
3. C#集合(13)
4. Autofac 组件、服务、自动装配 《第二篇》(13)
5. System.Web.Caching.Cache类 缓存 各种缓存依赖(12)

推荐排行榜

1. 逻辑数据库设计 - 单纯的树(递归关系数据)(40)
2. System.Web.Caching.Cache类 缓存 各种缓存依赖(33)
3. jQuery插件之ajaxFileUpload(33)
4. C#枚举(32)
5. MySQL函数(32)

SearchKeys	从数据库中查找名称相等的Keys的集合，特殊模式如h[ae]llo，仅英文有效。
GetRandomKey	随机获取一个已经被使用的Key
GetValue	根据Key获取值，只对string类型有效
GetValues	根据输入的多个Key获取多个值，支持泛型
GetTimeToLive	获取指定Key的项距离失效点的TimeSpan
GetSortedSetCount	获取已排序集合的项的数目，参数支持下标以及score筛选
ExpireEntryAt	根据指定的key设置一项的到期时间（DateTime）
ExpireEntryIn	根据指定的key设置一项的到期时间（TimeSpan）
FlushDb	清除本数据库的所有数据
FlushAll	清除所有数据库的所有数据
Shutdown	停止所有客户端，保存，关闭Redis服务
Save	保存数据DB文件到硬盘
SaveAsync	异步保存
RewriteAppendOnlyFileAsync	只在异步情况下将数据追加到服务器文件
WriteAll	
PublishMessage	将Message发送到指定的频道
StoreObject	
GetValuesMap	以键值对的方式返回值类型相同的多条数据，支持泛型与返回字符串。
字符串	
SetEntry	根据Key修改一个值，存在则覆盖。（只能设置字符串）
SetEntryIfNotExists	根据Key设置一个值，仅仅当Key不存在时有效，如Key已存在则不修改(只支持字符串)
SetEntryIfNotExists	根据Key设置一个值，返回旧值。
GetEntryType	根据Key获取当前存储的值是什么类型： None = 0 String = 1 List = 2 Set = 3

	SortedSet = 4 Hash = 5
--	---------------------------

3、内置集合

比如，IRedisClient支持在内部维护如下集合类型的数据：

- List<T>
- 排序的List<T>(.Net 4.0后的SortedSet)
- HashSet<T>

关于如下4种类型数据的操作：

方法	说明
AddItemToList	添加一个项到内部的List<T>
AddItemToSet	添加一个项到内部的HashSet<T>
AddItemToSortedSet	添加一个项到内部的排序List<T>，其中重载方法多了个score：排序值。优先按照score从小->大排序，否则按值小到大排序
AddRangeToList	一次过将参数中的List<T>中的多个值添加入内部的List<T>
AddRangeToSet	一次过将参数中的HashSet<T>中的多个值添加入内部的HashSet<T>
AddRangeToSortedSet	一次过将参数中的List<T>中的多个值添加到内部List<T>，重载方法的score表示排序值。
GetAllItemsFromList	获取指定ListId的内部List<T>的所有值
GetAllItemsFromSet	获取指定SetId的内部HashSet<T>的所有值
GetAllItemsFromSortedSet	获取指定ListId的内部已排序List<T>的所有值
GetAllItemsFromSortedSetDesc	获取指定ListId的内部已排序List<T>的所有值，不过获取的值是倒序排列后的。
GetRangeFromList	获取指定ListId的内部List<T>中指定下标范围的数据
GetRangeFromSortedSet	获取指定ListId的内部已排序List<T>中指定下标范围的数据
GetRangeFromSortedSet	获取指定SetId的内部HashSet<T>中指定下标范围的数据

GetRangeFromSortedSetByHighestScore	获取指定SetId的内部HashSet<T>中按照score由高->低排序后的分值范围的数据，并且支持skip、take
GetRangeFromSortedSetByLowestScore	同上，只不过是按score分值由低->高取一定范围内的数据
GetRangeFromSortedSetDesc	按倒序获取内部HashSet<T>的指定下标范围内的数据
GetRangeWithScoresFromSortedSet	与From相同，只不过获取的是键值对，数据中带分值score
GetRangeWithScoresFromSortedSetByHighestScore	同上
GetRangeWithScoresFromSortedSetByLowestScore	同上
GetRangeWithScoresFromSortedSetDesc	同上
GetAllWithScoresFromSortedSet	获取指定ListId的已排序的内部List<T>与其score
GetSortedItemsFromList	从指定ListId的List<T>中获取按指定排序的集合，支持Skip,Take
GetSortedEntryValues	从指定ListId的List<T>中获取经过排序指定开始位置与个数的项
RemoveAllFromList	移除指定ListId的内部List<T>
RemoveItemFromList	移除指定ListId的内部List<T>中第二个参数值相等的那一项
RemoveItemFromSet	从指定SetId的内部HashSet<T>中移除与第二个参数值相等的那一项
RemoveItemFromSortedSet	从指定ListId中已排序的内部List<T>中移除值相等的那一项
RemoveRangeFromSortedSet	从指定ListId已排序的List<T>中移除指定下标范围的项
RemoveRangeFromSortedSetByScore	从指定ListId已排序的List<T>中移除指定score范围的项
RemoveStartFromList	从指定ListId移除开头那一项

RemoveEndFromList	从指定ListId移除末尾那项
BlockingRemoveStartFromList	阻塞地从指定ListId移除开头那一项
BlockingRemoveStartFromLists	
RemoveEntry	根据传入的多个ListId，清除多个内部List<T>
RemoveAllLuaScripts	清除所有的 Lua 脚本缓存
RemoveEntryFromHash	
GetItemFromList	根据ListId和下标获取一项
GetItemIndexInSortedSet	根据ListId和值，获取内置的排序后的List<T>的下标
GetItemIndexInSortedSetDesc	同上，不过顺序相反
GetItemScoreInSortedSet	根据传入的ListId和值获取内置List<T>项的score
GetListCount	根据ListId，获取内置的List<T>的项数
GetSetCount	根据SetId，获取内置的HashSet<T>的项数
GetIntersectFromSets	从输入的多个HashSet<T>的Id中获取交集
GetUnionFromSets	从输入的多个HashSet<T>的Id中获取并集
GetRandomItemFromSet	从指定ListId的集合中获取随机项
StoreUnionFromSets	将多个HashSet<T>，合并为第一个参数中的一个HashSet<T>，第一个参数中的HashSet<T>原本可以不存在
StoreUnionFromSortedSets	将多个SortedSet<T>，合并为第一个参数中的一个SortedSet<T>，第一个参数中的SortedSet<T>原本可以不存在
StoreIntersectFromSets	将交集结果保存在第一个参数的集合中，对HashSet<T>作用
StoreIntersectFromSortedSets	将交集结果保存在第一个参数的集合中，对SortedSet<T>作用
EnqueueItemOnList	将一个元素存入指定ListId的List<T>的头部

st	
DequeueItemFromList	将指定ListId的List<T>末尾的那个元素出列，返回出列元素
BlockingDequeueItemFromList	将指定ListId的List<T>末尾的那个元素出列，区别是：会阻塞该List<T>，支持超时时间，返回出列元素
BlockingDequeueItemFromLists	
BlockingPopItemFromList	阻塞地将指定ListId的List<T>末尾的哪一个元素移除
BlockingPopItemFromLists	
BlockingPopAndPushItemBetweenLists	将第一个集合的元素移除并添加到第二个集合的头部，返回该元素，会同时阻塞两个集合
PopItemFromList	从指定ListId的List<T>末尾移除一项并返回
PopItemFromSet	从指定SetId的HashSet<T>末尾移除一项并返回
PopItemWithHighestScoreFromSortedSet	从指定SetId的HashSet<T>移除score最高的那一项
PopItemWithLowestScoreFromSortedSet	从指定SetId的HashSet<T>移除score最低的那一项
PopAndPushItemBetweenLists	将第一个集合的元素移除并添加到第二个集合的头部
SetContainsItem	判断指定SetId的HashSet<T>中是否包含指定的value(仅仅支持字符串)
SortedSetContainsItem	判断SortedSet是否包含一个键
TrimList	根据ListId裁剪内置集合，保留下去from->at之间(包含from于at)的元素，其余的裁去
IncrementItemInSortedSet	为指定ListId的集合中的value的分值score加上指定分值
SetItemInList	重新设置指定ListId和下标的value为指定值
PushItemToList	在指定ListId的内置List<T>中入列一个键值对，在末尾
PrependItemToList	将一个值插入到List<T>的最前面

t	
PrependRangeToLi st	一次性添加多个值到指定ListId的内置List<T> 中
GetDifferencesFro mSet	返回存在于第一个集合，但是不存在于其他集合 的数据。差集
StoreDifferencesFr omSet	将求差集的结果保存在第一个参数的集合中
MoveBetweenSets	将元素从一个集合移动到另一个集合的开头。(删 除与添加)

下面仅给出一个List<T>与HashSet<T>的示例：




```
//内部维护一个List<T>集合
RClient.AddItemToList("蜀国", "刘备");
RClient.AddItemToList("蜀国", "关羽");
RClient.AddItemToList("蜀国", "张飞");
List<string> ListString =
RClient.GetAllItemsFromList("蜀国");
foreach (string str in ListString)
{
    Response.Write(str);    //输出 刘备 关羽 张飞
}


RClient.AddItemToSet("魏国", "曹操");
RClient.AddItemToSet("魏国", "曹操");
RClient.AddItemToSet("魏国", "典韦");
HashSet<string> HashSetString =
RClient.GetAllItemsFromSet("魏国");
foreach (string str in HashSetString)
{
    Response.Write(str);    //输出 典韦 曹操
}
```



下面再给一个范围Range操作示例：



```
//内部维护一个List<T>集合
RClient.AddItemToSortedSet("蜀国", "刘备", 5);
RClient.AddItemToSortedSet("蜀国", "关羽", 2);
RClient.AddItemToSortedSet("蜀国", "张飞", 3);
IDictionary<String,double> DicString =
RClient.GetRangeWithScoresFromSortedSet("蜀国", 0, 2);
foreach (var r in DicString)
{
    Response.Write(r.Key + ":" + r.Value);    //输出
}
```



3、内置Hash

内部维护一个HashTable

方法	说明
SetEntryIn Hash	设置一个键值对入Hash表，如果哈希表的key存在则覆盖
SetEntryIn HashIfNot Exists	当哈希表的key未被使用时，设置一个键值对如Hash表
GetHashV alues	根据HashId获取多个改HashId下的多个值
GetValues FromHash	根据HashId和Hash表的Key获取多个值(支持多个key)
GetValueF romHash	根据HashId和Hash表的Key获取单个值
GetHashK eys	获取指定HashId下的所有Key
GetHashV alues	获取指定HashId下的所有值
GetHashC ount	获取指定HashId下的所有Key数量
HashConta insEntry	判断指定HashId的哈希表中是否包含指定的Key
Increment ValueInHa sh	将指定HashId的哈希表中的值加上指定值
StoreAsHa sh	将一个对象存入Hash(支持泛型)
GetFromH ash	根据Id从Hash表中取出对象(支持泛型)
SetRangeI nHash	通过IEnumerable<KeyValuePair<string, string>> 一次性设置多个值，当内部Hash的key不存在则添加，存在则覆盖

代码示例：

```
RCClient.SetEntryInHash("xxx", "key", "123");
List<KeyValuePair<string, string>> keyValuePairs = new
List<KeyValuePair<string, string>>();
KeyValuePair<string, string> kvp = new
KeyValuePair<string, string>("key", "1");
keyValuePairs.Add(kvp);
RCClient.SetRangeInHash("xxx", keyValuePairs);
```

4、Lua Script

从 Redis 2.6.0 版本开始，通过内置的 Lua 解释器，可以执行各种Lua脚本。IRedisClient支持执行Lua脚本，其供用于执行Lua脚本的方法如下：

方法	说明
LoadLuaScript	将一个脚本装入脚本缓存，但并不立即运行它
KillRunningLuaScript	停止正在运行的指定Id的脚本
ExecLuaAsInt	
ExecLuaAsList	
ExecLuaAsString	
ExecLuaShaAsInt	
ExecLuaShaAsList	
ExecLuaShaAsString	
HasLuaScript	判断Lua脚本是否在脚本缓存里
CalculateSha1	
WhichLuaScriptsExist s	

关于Lua脚本可以到这里去了解：

<http://www.cnblogs.com/ly4cn/archive/2006/08/04/467550.html>

5、事务

Redis中的事务

方法	说明
Watch	监视一个(或多个) key ，如果在事务执行之前这个(或这些) key 被其他命令所改动，那么事务将被打断。
UnWatch	取消 WATCH 命令对所有 key 的监视
AcquireLock	申请对一个Key加锁(期间其他对象不能访问)
CreateTransaction	创建一个事务，返回一个IRedisTransaction对象
CreateSubscription	创建一个订阅事件返回一个IRedisSubscription对象

on	
CreatePipeline	返回一个IRedisPipeline对象

分类: 性能 : 缓存

好文要顶

关注我

收藏该文







逆心

关注 - 31

粉丝 - 2395

15

0

+加关注

« 上一篇 : Redis 安装与简单示例 <第一篇>
» 下一篇 : Redis常用命令速查 <第二篇>

posted on 2014-02-27 17:52 逆心 阅读(38027) 评论(11) 编辑 收藏

Feedback

- #1楼 2014-06-18 18:31 凌晨一点

楼主，你从redis 获取中文会出现乱码吗？

支持(0) 反对(0)
- #2楼[楼主] 2014-06-19 08:31 逆心

@ 凌晨一点
遇到过，不过怎么弄好的忘记了。好像是配置问题。

支持(0) 反对(0)
- #3楼 2014-12-03 16:07 亲爱de小孩

楼主，如果有1主2从，master宕机后slave1升为主服务器，那怎么在c#里面把原来指向的master自动指向slave1呢

支持(0) 反对(0)
- #4楼[楼主] 2014-12-03 16:17 逆心

@ 亲爱de小孩
很遗憾，这只是看书学来的理论以及写了一些Demo，一直都没有实践的机会，所以你的问题，我知识有限，还帮不到你。

支持(0) 反对(0)
- #5楼 2015-01-23 11:35 IT_少年

挺有用，

支持(0) 反对(0)
- #6楼 2015-06-02 16:30 ljs0109

楼主,HashTable有没有方法一次性把所有key和value都取出来的方法吗

支持(0) 反对(0)
- #7楼 2015-10-22 21:20 SZW

Hashes 中的对象是否支持个别对象或者个别属性更新，而不是整个对象重置？
支持(0) 反对(0)

#8楼 2015-11-11 16:15 沐松

@ 亲爱de小孩
redis有哨兵

支持(0) 反对(0)

#9楼 2015-12-07 16:49 wujf

好全啊，赞

支持(0) 反对(0)

#10楼 2015-12-27 13:24 ThisIsTest

mark

支持(0) 反对(0)

#11楼 2016-02-18 18:37 long-2008

祝楼主幸福！很有用的文章！

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

- 【推荐】超50万C++/C#源码：大型实时仿真组态图形源码
- 【前端】SpreadJS表格控件，可嵌入系统开发的在线Excel

相关博文：

- [ServiceStack.Redis之IRedisClient<第三篇>](#)
- [ServiceStack.Redis之IRedisClient<第三篇>](#)
- [ServiceStack.Redis之IRedisClient<第三篇>](#)
- [ServiceStack.Redis之IRedisClient<第三篇>](#)
- [ServiceStack.Redis之IRedisClient<第三篇>](#)

最新新闻：

- 微软拆分操作系统的计划初现雏形
 - 5G让印度电信富豪头疼：买频谱太贵，不卖没前途
 - 股东提13项打击性骚扰反垄断促进多样化建议 被谷歌全否
 - 日经报道称苹果正考虑从中国转移15-30%的硬件产能
 - 沃尔玛递送大战小胜一筹 2018年申请无人机技术专利超亚马逊
- » 更多新闻...

Copyright @ 逆心

Powered by: .Text and ASP.NET

Theme by: .NET Monster

多肉|广州窗帘