窗体顶端

|  |
| --- |
| **LINQ TO OBJECTS**[[复制链接]](javascript:;) |
|  |  |

|  |  |
| --- | --- |
| **changfuguo**  头像   * 论坛积分[38 分](http://www.51aspx.com/Help/4) * 注册时间2007-12-24   源码贡献奖 | [楼主](http://bbs.51aspx.com/showtopic-7701.html###)  字体大小:  http://bbs.51aspx.com/images/groupicons/member.gif 发表于 2009-04-26 17:16 |[只看楼主](http://bbs.51aspx.com/showtopic.aspx?topicid=7701&forumpage=1&onlyauthor=1&posterid=3392)  [LINQ](http://bbs.51aspx.com/topictag-31.aspx) [头objects](http://bbs.51aspx.com/topictag-1139.aspx)  今天把学习LINQ TO ONJECT的东西分享一下，主要把书上东西拿出来，加上自己的一点疑问。 第一部分主要是LINQ 操作数组。第二部分操作字符串 操作数组主要有静态数组、动态数组、泛型列表、泛型排序列表、泛型双向列表、泛型队列、泛型堆栈、泛型哈希表、泛型排序字典、泛型通用集合、泛型绑定列表十一种。 一、操作数组 1） 静态数组 所谓静态数组指事先规定了数组的长度。比如  int[] ints = { 0,9,2,3,7,5,4,6,1,8 };  UserInfo[] ui = new UserInfo[5]; List<UserInfo> lui = new List<UserInfo>(10); 下面给出一个具体例子：  private void StringArrayQuery() { string[] strs = { "chang", "fu", "guo", "wo", "xiang", "ni" }; var result = from u in strs where u.IndexOf("an") > -1 select u; foreach (var v in result) { Response.Write(v + ","); } Response.Write("<br />"); }  查询结果 chang，xiang  2） 操作动态数组 ArryList  注意：数组类型不是基础类型，则需要进行类型转换  ArrayList users = new ArrayList();  for(int i = 1; i < 17; i++) { users.Add(new UserInfo(i % 5,"User0" + i.ToString(),"User0" + i.ToString() + "@web.com")); } var values = from UserInfo u in users where u.Username.Length > 6 orderby u.ID descending, u.Username select u; ///使用Cast函数转换数据类型 var valuesOther = from u in users.Cast<UserInfo>() where u.ID > 2 orderby u.ID,u.Username descending select u; ///显示查询结果 foreach(var v in values) { Response.Write("(" + v.ID.ToString() + "," + v.Username + ")</br>"); } Response.Write("<br />"); foreach(var v in valuesOther) { Response.Write("(" + v.ID.ToString() + "," + v.Username + ")</br>"); } Response.Write("<br />"); 运行结果为：  (4,User014) (3,User013) (2,User012) (1,User011) (1,User016) (0,User010) (0,User015)  (3,User08) (3,User03) (3,User013) (4,User09) (4,User04) (4,User014) 3） 操作泛型列表List<T> 注意：泛型列表通过索引访问，其中元素不是排序的，，可以包含重复元素或NULL，可以使用相等比较器，也可以使用排序比较器。 其中对于每一个类型T来说要实现排序方法必须重写比较的接口。  ///构建数据源  List<UserInfo> users = new List<UserInfo>(); for(int i = 1; i < 10; i++) { users.Add(new UserInfo(i % 2,"User0" + i.ToString(),"User0" + i.ToString() + "@web.com")); } ///查询泛型数组 var values = from u in users orderby u.ID, u.Username select u; ///显示查询结果 foreach(var v in values) { Response.Write("(" + v.ID.ToString() + "," + v.Username + ")</br>"); } Response.Write("<br />"); 运行结果如下  ///构建数据源 List<UserInfo> users = new List<UserInfo>(); for(int i = 1; i < 10; i++) { users.Add(new UserInfo(i % 2,"User0" + i.ToString(),"User0" + i.ToString() + "@web.com")); } ///查询泛型数组 var values = from u in users orderby u.ID, u.Username select u; ///显示查询结果 foreach(var v in values) { Response.Write("(" + v.ID.ToString() + "," + v.Username + ")</br>"); } Response.Write("<br />");  运行结果如下：  (0,User02) (0,User04) (0,User06) (0,User08) (1,User01) (1,User03) (1,User05) (1,User07) (1,User09)  4） 泛型排序列表StordList<Tkey，Tvalue> 泛型排序是“键/值”对组成，其中键必须是唯一的，不能修改不能为空，可以按照键值排序，可以比较和排序。可以根据键来获取指向的值。  ///构建数据源  SortedList<int,UserInfo> users = new SortedList<int,UserInfo>(); for (int i = 1; i < 10; i++) {  users.Add(i, new UserInfo(i,"User0" + i.ToString(),"User0" + i.ToString() + "@web.com")); } ///查询排序泛型列表 var values = from u in users where u.Value.ID > 5 && u.Value.ID < 10 orderby u.Value.Username descending select u; ///显示查询结果 foreach (var v in values) { Response.Write("(" + v.Value.ID.ToString() + "," + v.Value.Username + ")</br>");    }  Response.Write("<br />"); 运行结果  (9,User09) (8,User08) (7,User07) (6,User06)      5） 泛型双向列表 LinkList<T>    泛型双向链表可以通过当前元素直接访问其后续和前驱元素。如果为空则其First 和Last都为空    AddBefore，AddAfter通过这两个属性添加    代码入下    ///构建数据源    LinkedList<UserInfo> ints = new LinkedList<UserInfo>();    UserInfo u =new UserInfo(0,"chang0","0@qq.com");    ints.AddFirst(u);      SortedList<int, UserInfo> sl = new SortedList<int, UserInfo>();    for (int i = 1; i < 7; i++)    {    sl.Add(i , new UserInfo(i%2, "chang0" + i.ToString(), i.ToString() + "@QQ.com"));    if (i == 1)    {    ints.AddAfter(ints.Find(u), sl.Values[i - 1]);    }    else    {    ints.AddAfter(ints.Find( sl.Values[i-2]),sl.Values[i-1]);      }    }    ///查询泛型双向链表    var values = from i in ints    orderby i.Username descending    select i;    ///显示查询结果    foreach (var v in values)    {    Response.Write(v.Username + "=>");    }    Response.Write("<br />");    运行结果如下：    chang06=>chang05=>chang04=>chang03=>chang02=>chang01=>chang0=> |
| [分享](http://bbs.51aspx.com/misc.aspx?action=emailfriend&tid=7701) [转发](javascript:void(0)) |
|  |
|  | [TOP](http://bbs.51aspx.com/showtopic-7701.html) |
|  |  |

|  |  |
| --- | --- |
| **changfuguo**  头像   * 论坛积分[38 分](http://www.51aspx.com/Help/4) * 注册时间2007-12-24   源码贡献奖 | [沙发](http://bbs.51aspx.com/showtopic-7701.html###)  字体大小:  http://bbs.51aspx.com/images/groupicons/member.gif 发表于 2009-04-26 17:17 |[只看楼主](http://bbs.51aspx.com/showtopic.aspx?topicid=7701&forumpage=1&onlyauthor=1&posterid=3392)  **LINQ TO OBJECTS**  6） 泛型队列 Queue<T>    队列是先进先出的线性表，类似两头透气的管，这头先进那头先出。允许有重复元素并且为空Enqueue(T)进站操作，出站操作Dequeue（）返回一个T，    例子如下    ///构建数据源      Queue<UserInfo> users = new Queue<UserInfo>();     for (int i = 1; i < 10; i++)     {          users.Enqueue(new UserInfo(i % 2,"User0" + i.ToString(),"User0" + i.ToString() + "@web.com"));     }     ///查询泛型队列          var values = from u in users      orderby u.ID,u.Username      select u;     ///显示查询结果     foreach (var v in values)     {     Response.Write("(" + v.ID.ToString() + "," + v.Username + ")</br>");     }     Response.Write("<br />");    结果如下    (0,User02) (0,User04) (0,User06) (0,User08) (1,User01) (1,User03) (1,User05) (1,User07) (1,User09)    7） 泛型堆栈是一个先进后出的线性表，类似一个这头进这头出的管子。除此之外性质和QUEUE 一样。操作主要有Push（T）将T压入堆栈，Pop（）饭后盏顶元素 代码如下：  ///构建数据源      Stack<UserInfo> users = new Stack<UserInfo>();     for (int i = 1; i < 10; i++)     {          users.Push(new UserInfo(i % 2,"User0" + i.ToString(),"User0" + i.ToString() + "@web.com"));     }     ///查询泛型堆栈     var values = from u in users            select u;     ///显示查询结果     foreach (var v in values)     {     Response.Write("(" + v.ID.ToString() + "," + v.Username + ")</br>");     }    Response.Write("<br />"); 结果如下  (1,User09) (0,User08) (1,User07) (0,User06) (1,User05) (0,User04) (1,User03) (0,User02) (1,User01) 和正常的出站顺序一致。 8） 操作哈希集合，提供高性能集合，如并集交集等。不考虑排序、不包含重复元素。  HashSet<UserInfo> users = new HashSet<UserInfo>();  for (int i = 1; i < 10; i++) { users.Add(new UserInfo(i % 2,"User0" + i.ToString(),"User0" + i.ToString() + "@web.com")); } ///查询泛型哈希表 var values = from u in users //orderby u.ID,u.Username select u; ///显示查询结果 foreach (var v in values) Response.Write("(" + v.ID.ToString() + "," + v.Username + ")</br>"); } Response.Write("<br />"); (1,User01) 结果 (0,User02) (1,User03) (0,User04) (1,User05) (0,User06) (1,User07) (0,User08) (1,User09) 9）操作泛型字典 Dictionary<Tkey,TValue>,键值对的集合，键不能修改不能为空唯一。值可以为空和sortlist差不多 代码如下  Dictionary<int,UserInfo> users = new Dictionary<int,UserInfo>(); for (int i = 1; i < 10; i++) users.Add(i,new UserInfo(i,"User0" + i.ToString(),"User0" + i.ToString() + "@web.com")); } ///查询排序泛型字 var values = from u in users orderby u.Value.ID,u.Value.Username select u; ///显示查询结果 foreach (var v in values) { Response.Write("(" + v.Value.ID.ToString() + "," + v.Value.Username + ")</br>"); } Response.Write("<br />"); 需要通过键->值->T-> 属性逐步得到 结果为 (1,User01) (0,User02) (1,User03) (0,User04) (1,User05) (0,User06) (1,User07) (0,User08) (1,User09) 10）通用集合 Collection<T> 和List差不多但是可以通过扩展他事先更复杂的操作。 具体操作部写了.. 11)泛型绑定列表 支持数据绑定的泛型集合，提供IbindingList的具体实现时创作双向绑定机制的基类，通过ADDnew()支持工厂创建实例。                （本人不是很明白说的这啥意思） 12 ）泛型排序列表SortedDictionary<Tkey,Tvalue>  ///构建数据源 SortedDictionary<int,UserInfo> users = new SortedDictionary<int,UserInfo>(); for (int i = 1; i < 10; i++) { users.Add(i,new UserInfo(i,"User0" + i.ToString(),"User0" + i.ToString() + "@web.com")); } ///查询排序泛型字典 var values = from u in users orderby u.Value.ID,u.Value.Username select u; ///显示查询结果 foreach (var v in values) { Response.Write("(" + v.Value.ID.ToString() + "," + v.Value.Username + ")</br>"); } Linq我个人觉得在速度上比ADO要快很多，并且操纵简单，但是作为一种新的技术要看用的合适不合适。还有我这里发现两个问题： 第一个操作存储过程的时候，如果存储过程 有两个 查询 Select \* from a Select \*from b 在LINQ TO SQL 类中自动生成如下 [Function(Name="dbo.Pr\_GetUsersAndRoles")] public ISingleResult<Pr\_GetUsersAndRolesResult> Pr\_GetUsersAndRoles() { IExecuteResult result = this.ExecuteMethodCall(this, ((MethodInfo)(MethodInfo.GetCurrentMethod()))); return ((ISingleResult<Pr\_GetUsersAndRolesResult>)(result.ReturnValue)); } 但是调用的时候  LinqDBDataContext db = new LinqDBDataContext(LinqSystem.LinqDBConnectionString);  ///获取分类结果 System.Data.Linq.IMultipleResults result = db.Pr\_GetUsersAndRoles().  ///显示用户信息 foreach (UserInfo u in result.GetResult<UserInfo>()) { Response.Write("ID:" + u.ID.ToString() + "，"); Response.Write("Username:" + u.Username + "，"); Response.Write("Email:" + u.Email + "。<br />"); } ///显示角色信息 foreach (Role r in result.GetResult<Role>()) { Response.Write("ID:" + r.ID.ToString() + "，"); Response.Write("RoleName:" + r.RoleName + "。<br />"); } 就会提示出错  CS0266: 无法将类型“System.Data.Linq.ISingleResult<Pr\_GetUsersAndRolesResult>”隐式转换为“System.Data.Linq.IMultipleResults”。存在一个显式转换(是否缺少强制转换?) 如果将存储过程中返回的值改为如下 public IMultipleResults<Pr\_GetUsersAndRolesResult> Pr\_GetUsersAndRoles() { IExecuteResult result = this.ExecuteMethodCall(this, ((MethodInfo)(MethodInfo.GetCurrentMethod()))); return ((IMultipleResults<Pr\_GetUsersAndRolesResult>)(result.ReturnValue)); 也会出现错误有没遇到过，，请拍砖？ 第二个问题就是如何选择不同的集合，，就是什么情况下用什么集合？这个问题我觉得比较重要，很简单的例子比如哈希表和泛型字典的区别在吗？请拍砖,请高手指点，其中大部分是书上例子，小部分作了改动。 |
|

窗体底端