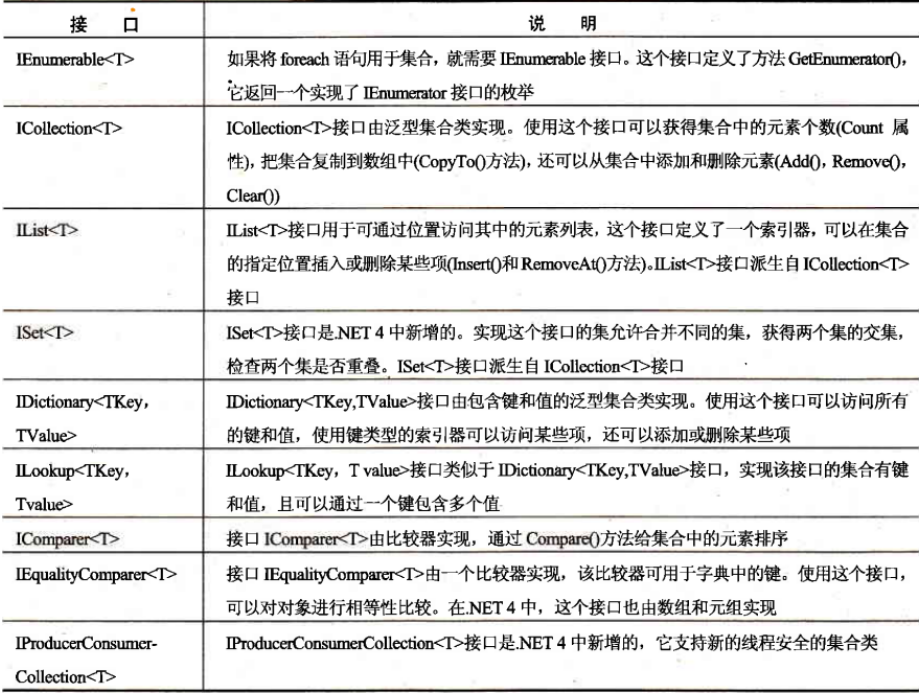
大多数集合都在System.Collections，System.Collections.Generic两个命名空间。其中System.Collections.Generic专门用于泛型集合。

针对特定类型的集合类型位于System.Collections.Specialized;命名空间；

线程安全的集合类位于System.Collections.Concurrent;命名空间。

下面是集合和列表实现的接口如下：



一、列表

[Serializable]

[DebuggerTypeProxy(typeof(Mscorlib\_CollectionDebugView<>))]

[DebuggerDisplay("Count = {Count}")]

public class List<T> : IList<T>, ICollection<T>, IEnumerable<T>, IList, ICollection, IEnumerable

从这个可以看出，泛型集合List<T>实现了这么多接口，具体接口的信息可以通过工具查看。

using System;

using System.Collections.Generic;

namespace ConsoleApplication1

{

public class Program

{

static void Main(string[] args)

{

List<String> list = new List<string>();

list.Add("张三");

list.Add("李四");

list.Add("王五");

list.Add("田六");

list.Add("赵七");

for (int i = 0; i < list.Count; i++)

{

Console.WriteLine("for循环：" + i.ToString() + "=" + list[i]);

}

list.RemoveAt(0);

foreach (String item in list)

{

Console.WriteLine("foreach迭代：" + item);

}

list.AddRange(new String[] { "Hello1", "Hello2", "Hello3" });

list.ForEach(Print);

Console.Read();

}

private static void Print(String item)

{

Console.WriteLine("ForEach：" + item);

}

}

}

 二、队列

队列先进先出，一头进一头出，用Queue<T>实现

[Serializable]

[DebuggerTypeProxy(typeof(System\_QueueDebugView<>))]

[ComVisible(false)]

[DebuggerDisplay("Count = {Count}")]

public class Queue<T> : IEnumerable<T>, ICollection, IEnumerable

可以看出队列实现了集合的接口，迭代的接口

using System;

using System.Collections.Generic;

namespace ConsoleApplication1

{

public class Program

{

static void Main(string[] args)

{

Queue<String> queue = new Queue<string>();

//进队

queue.Enqueue("张三");

queue.Enqueue("李四");

queue.Enqueue("王五");

queue.Enqueue("田六");

queue.Enqueue("赵七");

foreach (String item in queue)

{

Console.WriteLine("foreach迭代：" + item);

}

//出队

while (queue.Count > 0)

{

Console.WriteLine("出队：" + queue.Dequeue());

}

Console.Read();

}

}

}

 三、栈

栈：从同一边先进后出，用Stack<T>实现

[DebuggerDisplay("Count = {Count}")]

[DebuggerTypeProxy(typeof(System\_StackDebugView<>))]

[ComVisible(false)]

public class Stack<T> : IEnumerable<T>, ICollection, IEnumerable

栈也是实现了集合接口与迭代接口的

using System;

using System.Collections.Generic;

namespace ConsoleApplication1

{

public class Program

{

static void Main(string[] args)

{

Stack<String> stack = new Stack<string>();

//进栈

stack.Push("张三");

stack.Push("李四");

stack.Push("王五");

stack.Push("田六");

stack.Push("赵七");

foreach (String item in stack)

{

Console.WriteLine("foreach迭代：" + item);

}

//出栈

while (stack.Count > 0)

{

Console.WriteLine("出栈：" + stack.Pop());

}

Console.Read();

}

}

}

 四、链表

LinkedList是一个双向链表，链表有个有点，就是在链表中间插入、删除元素很快，但是查找中间与末尾的元素很慢，需要一个节点一个节点的去找。

[Serializable]

[DebuggerTypeProxy(typeof(System\_CollectionDebugView<>))]

[DebuggerDisplay("Count = {Count}")]

[ComVisible(false)]

public class LinkedList<T> : ICollection<T>, IEnumerable<T>, ICollection, IEnumerable, ISerializable, IDeserializationCallback

由此可见链表也是有集合的特性的，可以迭代，同时还有链表的特性

using System;

using System.Collections.Generic;

namespace ConsoleApplication1

{

public class Program

{

static void Main(string[] args)

{

LinkedList<String> lList = new LinkedList<string>();

LinkedListNode<String> node = new LinkedListNode<string>("root");

lList.AddFirst(node);

node = lList.AddAfter(node, "张三");

node = lList.AddAfter(node, "李四");

node = lList.AddAfter(node, "王五");

node = lList.AddAfter(node, "田六");

node = lList.AddAfter(node, "赵七");

foreach (String item in lList)

{

Console.WriteLine("foreach迭代：" + item);

}

node = lList.First;

Console.WriteLine("第一个元素：" + node.Value);

node = lList.Last;

Console.WriteLine("最后一个元素：" + node.Value);

Console.Read();

}

}

}

 五、有序列表

SortedList采用键-值对存储，键不能重复，并且会根据key进行排序

[Serializable]

[DebuggerTypeProxy(typeof(System\_DictionaryDebugView<,>))]

[DebuggerDisplay("Count = {Count}")]

[ComVisible(false)]

public class SortedList<TKey, TValue> : IDictionary<TKey, TValue>, ICollection<KeyValuePair<TKey, TValue>>, IEnumerable<KeyValuePair<TKey, TValue>>, IDictionary, ICollection, IEnumerable

可以看出SortedList不仅具有字典的特性，还有集合，迭代的功能

using System;

using System.Collections.Generic;

namespace ConsoleApplication1

{

public class Program

{

static void Main(string[] args)

{

//Key必须唯一，如果不唯一可以考虑Lookup<TKey,TElement>

SortedList<int, String> sList = new SortedList<int, string>();

sList.Add(100, "张三");

sList.Add(21, "李四");

sList.Add(13, "王五");

sList.Add(44, "田六");

sList.Add(35, "赵七");

foreach (KeyValuePair<int, String> item in sList)

{

Console.WriteLine("key=" + item.Key.ToString() + "；value=" + item.Value);

}

Console.Read();

}

}

}

 六、字典

字典是很复杂的数据结构，允许通过key来查找值，字典可以自由添加、删除元素，没有集合由于移动元素导致的开销。

[Serializable]

[DebuggerTypeProxy(typeof(Mscorlib\_DictionaryDebugView<,>))]

[DebuggerDisplay("Count = {Count}")]

[ComVisible(false)]

public class Dictionary<TKey, TValue> : IDictionary<TKey, TValue>, ICollection<KeyValuePair<TKey, TValue>>, IEnumerable<KeyValuePair<TKey, TValue>>, IDictionary, ICollection, IEnumerable, ISerializable, IDeserializationCallback

可以看出字典也具有集合的特性，可以迭代

using System;

using System.Collections.Generic;

namespace ConsoleApplication1

{

public class Program

{

static void Main(string[] args)

{

//Key必须唯一

Dictionary<int, String> dict = new Dictionary<int, string>();

dict.Add(11, "张三");

dict.Add(1, "李四");

dict.Add(2, "王五");

dict.Add(16, "田六");

dict.Add(12, "赵七");

foreach (KeyValuePair<int, String> item in dict)

{

Console.WriteLine("key=" + item.Key.ToString() + "；value=" + item.Value);

}

Console.Read();

}

}

}

 说到字典，顺便谈一下有序字典，与有序列表对应；SortedDictionary，SortedList，SortedSet

会根据Key进行排序

using System;

using System.Collections.Generic;

namespace ConsoleApplication1

{

public class Program

{

static void Main(string[] args)

{

//Key必须唯一

SortedDictionary<int, String> dict = new SortedDictionary<int, string>();

dict.Add(11, "张三");

dict.Add(1, "李四");

dict.Add(2, "王五");

dict.Add(16, "田六");

dict.Add(12, "赵七");

foreach (KeyValuePair<int, String> item in dict)

{

Console.WriteLine("key=" + item.Key.ToString() + "；value=" + item.Value);

}

Console.Read();

}

}

}

 七、集

集(Set)：包含不重复元素，常用HashSet，SortedSet

[Serializable]

[DebuggerDisplay("Count = {Count}")]

[DebuggerTypeProxy(typeof(HashSetDebugView<>))]

public class HashSet<T> : ISerializable, IDeserializationCallback, ISet<T>, ICollection<T>, IEnumerable<T>, IEnumerable

[Serializable]

[DebuggerTypeProxy(typeof(SortedSetDebugView<>))]

[DebuggerDisplay("Count = {Count}")]

public class SortedSet<T> : ISet<T>, ICollection<T>, IEnumerable<T>, ICollection, IEnumerable, ISerializable, IDeserializationCallback

using System;

using System.Collections.Generic;

namespace ConsoleApplication1

{

public class Program

{

static void Main(string[] args)

{

HashSet<String> hSet = new HashSet<string>();

hSet.Add("张三");

hSet.Add("李四");

hSet.Add("王五");

hSet.Add("田六");

hSet.Add("赵七");

foreach (String item in hSet)

{

Console.WriteLine("foreach迭代：" + item);

}

Console.Read();

}

}

}

using System;

using System.Collections.Generic;

namespace ConsoleApplication1

{

public class Program

{

static void Main(string[] args)

{

SortedSet<String> hSet = new SortedSet<string>();

hSet.Add("张三");

hSet.Add("李四");

hSet.Add("王五");

hSet.Add("田六");

hSet.Add("赵七");

foreach (String item in hSet)

{

Console.WriteLine("foreach迭代：" + item);

}

Console.Read();

}

}

}

性能比较：

