集合查询扩展方法

北京理工大学计算机学院
金旭亮

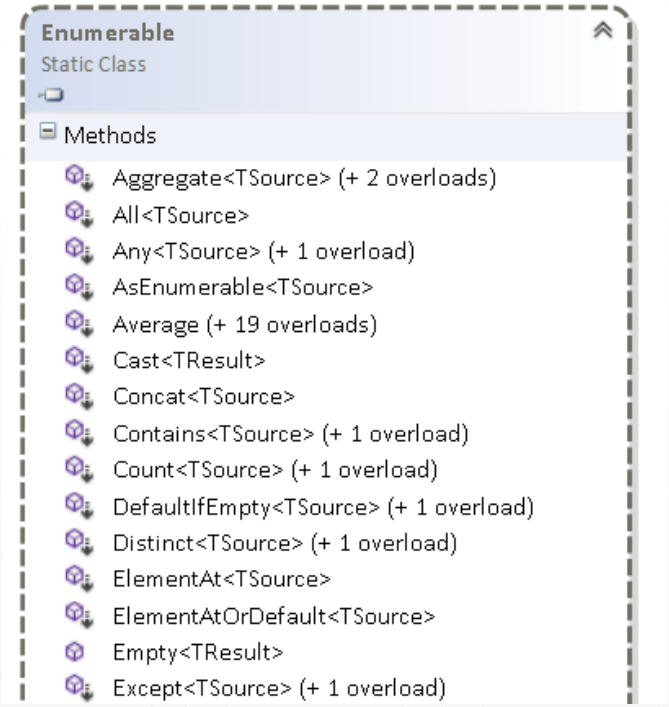
从.NET 3.5开始，.NET基类库中提供了一组扩展方法，为实现了IEnumerable<T>和IQueryable<T>接口的对象提供各种数据查询和处理功能。

这组扩展方法，实际上是另一个.NET重量级技术——“LINQ（语言集成的查询）”的基础，在实际开发中用得很广。

```
public static IEnumerable<TSource> Where<TSource>(
        this IEnumerable<TSource> source, Func<TSource, bool> predicate);
```

针对IEnumerable接口的扩展方法。

广泛使用委托作为方法参数，在调用时，经常使用Lambda表达式作为方法实参。



各种数据查询扩展方法集中于**System.Linq**命名空间的**Enumerable**静态类中，有近百个之多，实现了开发中常用的基本数据查询与处理功能。

这组扩展方法，有时被称为“**标准查询运算符（Standard Query Operators）**”。

常用集合查询扩展方法选讲

示例项目：UseQueryMethods

“筛选”指从某对象集合中选出那些满足特定条件的对象操作。它又称为“选择”，可以通过通过Where扩展方法实现

```
public static IEnumerable<TSource> Where<TSource>(
    this IEnumerable<TSource> source, Func<TSource, bool> predicate);
```

```
//创建一个包容[1,100]内整数的数组
IEnumerable<int> nums = Enumerable.Range(1, 100);
//使用Where扩展方法筛选出5的倍数，构成一个集合返回
var numList = nums.Where((num) => num % 5 == 0);
```

interface System.Collections.Generic.IEnumerable<out T>
Exposes the enumerator, which supports a simple iteration over a collection of a specified type.

T is System.Int32

很多查询扩展方法返回IEnumerable<T>的集合，可以进一步地“级联”其他查询扩展方法

数据的投影与转换

- 所谓“**投影（Project）**”，指把某对象集合中的对象的部分属性抽取出来进行处理。
- “**数据转换**”：指将某对象集合中的感兴趣的对象的部分属性转换为另一种类型的对象集合。
- 这两种操作都可以用**Select**运算符实现。

```
public static IEnumerable<TResult> Select<TSource, TResult>(
        this IEnumerable<TSource> source,
        Func<TSource, TResult> selector
)
```


在开发中，经常会使用Lambda表达式返回一个包容了相应数据的匿名对象

Select方法使用示例

```
//fileList的类型为：string[]
var fileList = Directory.GetFiles("c:\\", "*.*");
//files的类型为：IEnumerable<FileInfo>
var files = fileList.Select(
    file => new FileInfo(file));
```

找出C盘根目录中的所有文件，注意Lambda表达式把string对象转换为了FileInfo对象

```
//infos是匿名对象的集合，此匿名对象包容两个属性：文件名和文件大小
var infos=files.Select(info=>new {FileName=info.Name,
    FileLength=info.Length});
//输出结果
foreach (var info in infos)
{
    Console.WriteLine("{0}:{1}字节",info.FileName,
        info.FileLength);
}
```

抽取出FileInfo对象的文件名和文件大小两个属性，构建一个匿名对象。

数据排序可以使用OrderBy（升序）或OrderByDescending（降序）方法实现

```
public static IOrderedEnumerable<TSource>
        OrderBy<TSource, TKey>(
        this IEnumerable<TSource> source,
        Func<TSource, TKey> keySelector
)
```

指明从TSource中提取哪个属性作为排序的依据。
注意，**排序的属性必须是可比较大小的**，如果这个属性引用一个对象，则要求此对象实现IComparable接口（或其泛型版本）

```
//先把String对象转换为FileInfo对象，之后再按文件大小降序排列
var fileInfos = Directory.GetFiles("C:\\")
    .Select(file => new FileInfo(file))
    .OrderByDescending(fileInfo => fileInfo.Length);
```

应用实例

编程练习——多属性排序

除了OrderBy（OrderByDescending）之外，.NET基类库还提供了ThenBy（ThanByDescending）方法，可以使用它实现多属性排序，比如对文件集合先按文件大小，再按文件名进行排序。

请编写练习代码实现这一功能。

针对于集合，.NET基类库中提供了一组方法，实现了集合代数的标准运算，例如：

- **Distinct**：从集合中移除重复值。
- **Intersect**：交集。交集是指同时出现在两个集合中的元素。
- **Union**：并集。并集合并两个集合，如果有相同的元素，则只保留一个。
- **Except**：差集。两个集合进行差集运算时，返回属于第一个集合但不属于第二个集合的元素。
- ……

.NET同样提供了一些方法，完成了对集合元素的简单的统计功能，比如求集合中的**最大元素（Max）**、**平均值（Average）**等，用法请参看本讲示例。

编程练习

编写一个程序，自动生成一个由随机数构成的数组，使用扩展方法求其总和，平均值，最大值，最小值等统计数值，熟悉并掌握.NET基类库中常用扩展方法的用法。

提示：使用Random类生成随机数。

小结

.NET基类库所提供针对IEnumerable接口的扩展方法，实现了最常用的数据查询与处理操作。

在实际开发中，我们还可以依据具体场景，开发新的扩展方法。

用好这些扩展方法，能帮助我们写出简洁的代码。