

您的潜力，我们的动力



C# 3.0 锐利体验系列课程(3):

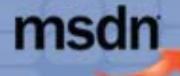
查询表达式 LINQ (1)

李建忠

jianzhong.lee@gmail.com

祝成科技 高级培训讲师

www.softcompass.com



MSDN Webcasts

您的潜力，我们的动力



C# 3.0语言主要增强

- 隐含类型局部变量
- 扩展方法
- 对象与集合初始化器
- 匿名类型
- Lambda表达式
- 查询表达式 LINQ (1)
- 表达式树

LINQ简介

- OO（面向对象）以外的疆域：信息的访问与整合。关系数据库与XML为其中的典型应用。
- .NET Language Integrated Query (LINQ): 不采用特定於关系数据库或者XML的专有方案，而采用通用方案来解决各种信息源的访问与整合问题。
- 在LINQ中，查询成为编程语言的一个组成部分，这使得查询表达式可以得到很好的编译时语法检查，丰富的元数据，智能感知等强类型语言的好处。

初识LINQ表达式

```
class app {
    static void Main() {
        string[] names = { "Burke", "Connor", "Frank",
                           "Everett", "Albert", "George",
                           "Harris", "David" };

        IEnumerable<string> query = from s in names
                                      where s.Length == 5
                                      orderby s
                                      select s.ToUpper();
        foreach (string item in query)
            Console.WriteLine(item);
    }
}
```

查询表达式解析（1）

```
IEnumerable<string> query = from s in names  
    where s.Length == 5  
    orderby s  
    select s.ToUpper();
```

在语义上等同于如下“方法风格（基于方法）的查询”：

```
IEnumerable<string> query = names  
    .Where(s => s.Length == 5)  
    .OrderBy(s => s)  
    .Select(s => s.ToUpper());
```

查询表达式解析（2）

注意其中的参数为lambda 表达式，类似于如下委托：

```
Func<string, bool> filter = delegate (string s) {  
    return s.Length == 5;};  
Func<string, string> extract = delegate (string s) {  
    return s; };  
Func<string, string> project = delegate (string s) {  
    return s.ToUpper(); };  
  
IEnumerable<string> query = names.Where(filter)  
    .OrderBy(extract)  
    .Select(project);
```

查询操作符与扩展方法解析（1）

查询操作符是LINQ中的另外一项重要设施，LINQ使用扩展方法来定义查询操作符，例如where操作符：

```
namespace System.Linq {  
    public static class Enumerable {  
        public static IEnumerable<T> Where<T>(  
            this IEnumerable<T> source, Func<T, bool> predicate) {  
  
            foreach (T item in source)  
                if (predicate(item))  
                    yield return item;  
        }  
    }  
}
```

查询操作符与扩展方法解析（2）

普通的方式来调用扩展方法：

```
IEnumerable<string> query = Enumerable.Where(names,  
    s => s.Length < 6);
```

C#语言允许我们使用如下的方式来调用扩展方法：

```
IEnumerable<string> query = names.Where(s => s.Length < 6);
```

您的潜力，我们的动力



更多查询操作符

Code Example

您的潜力，我们的动力

Microsoft®
微软(中国)有限公司

Microsoft®

msdn

MSDN Webcasts