

易百教程

- [所有教程](#)
 - [Java技术](#)
 - [Web技术](#)
 - [脚本语言](#)
 - [数据库](#)
 - [高级语言](#)
 - [其它技术](#)
 - [查看所有教程](#)
- [Java技术](#)
 - [Java教程](#)
 - [Java设计模式](#)
 - [JSP教程](#)
 - [JDBC教程](#)
 - [Java XML教程](#)
 - [JUnit教程](#)
 - [Struts2教程](#)
 - [Maven教程](#)
 - [java实例教程](#)
 - [MyBatis教程](#)
 - [Spring教程](#)
 - [Spring MVC教程](#)
 - [Gradle教程](#)
- [数据库](#)
 - [MySQL教程](#)
 - [CouchDB教程](#)

- [Cassandra教程](#)
- [MariaDB教程](#)
- [PL/SQL教程](#)
- [PostgreSQL教程](#)
- [MongoDB教程](#)
- [SQLite教程](#)
- [DB2教程](#)
- [Redis教程](#)
- [Memcached教程](#)
- [Access教程](#)
- [SQL教程](#)
- [SQL Server教程](#)
- [脚本语言](#)
 - [WxPython教程](#)
 - [Ruby教程](#)
 - [PHP教程](#)
 - [Python教程](#)
 - [Lua教程](#)
 - [Tcl教程](#)
 - [Tk教程](#)
 - [NumPy教程](#)
 - [Python3教程](#)
 - [PHP7教程](#)
- [极客文章](#)

[登录](#)



- [LinQ教程](#)

- [LINQ环境安装设置](#)
- [LINQ查询运算符](#)
- [LINQ过滤运算符](#)
- [LINQ Join操作](#)
- [LINQ投影操作](#)
- [LINQ排序运算符](#)
- [LINQ分组操作](#)
- [LINQ转换操作](#)
- [LINQ级联](#)
- [LINQ聚合](#)
- [LINQ量词操作](#)
- [LINQ分区操作符](#)
- [LINQ生成操作](#)
- [LINQ Set操作](#)
- [LINQ相等](#)
- [LINQ元素操作符](#)
- [LINQ SQL](#)
- [LINQ对象](#)
- [LINQ Dataset \(数据集\)](#)
- [LINQ XML](#)
- [LINQ实体](#)
- [LINQ Lambda表达式](#)
- [LINQ ASP.Net](#)

LINQ Lambda表达式

¥ 我要打赏 [LinQ教程](#) 作者：自主创新 评论：0 条 **Java技术QQ群**：[227270512](#)

术语“Lambda表达式”是来自于“拉姆达”演算这又是施加用于定义功能的数学符号及其名称。lambda表达式作为LINQ式的可执行部分转换的方式在运行时的逻辑，因此它可以传递到数据源方便。但是，lambda表达式不仅仅局限于查找应用LINQ而已。

这些表达式由下面的语法表示：

(输入参数) => 表达式或语句块

下面是lambda表达式的一个例子

$y \Rightarrow y * y$

上述表达式指定一个参数y和y的值的平方。然而，这是不可能的到这种形式来执行一个lambda表达式。在C#中的lambda表达式的例子如下所示。

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace lambdaexample
{
    class Program
    {
        delegate int del(int i);
        static void Main(string[] args)
        {
            del myDelegate = y => y * y;
            int j = myDelegate(5);
            Console.WriteLine(j);
            Console.ReadLine();
        }
    }
}
```

VB

```
Module Module1
    Private Delegate Function del(ByVal i As Integer) As Integer
    Sub Main(ByVal args As String())
        Dim myDelegate As del = Function(y) y * y
        Dim j As Integer = myDelegate(5)
        Console.WriteLine(j)
        Console.ReadLine()
    End Sub
End Module
```

当C#或VB的上述代码被编译和执行时，它产生了以下结果：

25

表达Lambda

如在lambda表达式的上面所示的语法表达是在右手侧，这些也被称为lambda表达式。

异步Lambdas

通过使用异步关键字结合异步处理创建lambda表达式被称为异步lambda表达式。下面是异步lambda的一个例子。

```
Func<Task<string>> getWordAsync = async() => "hello" ;
```

Lambda的标准查询操作

查询运算符内的lambda表达通过在需要时相同的评估计算，并不断地作用于各输入序列中的元素，而不是整个序列。开发人员允许Lambda表达式到自己的逻辑转换成标准查询操作。在下面的例子中，开发人员使用的“Where”运算符，通过利用lambda表达式的回收，从给出的列表中奇数值。

C#

```
//Get the average of the odd Fibonacci numbers in the series...
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace lambdaexample
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] fibNum = { 1, 1, 2, 3, 5, 8, 13, 21, 34 };
            double averageValue = fibNum.Where(num => num % 2 == 1).Average();
            Console.WriteLine(averageValue);
            Console.ReadLine();
        }
    }
}
```

VB

```
Module Module1
    Sub Main()
        Dim fibNum As Integer() = {1, 1, 2, 3, 5, 8, 13, 21, 34}
        Dim averageValue As Double = fibNum.Where(Function(num) num Mod 2 = 1).Average()
        Console.WriteLine(averageValue)
        Console.ReadLine()
    End Sub
End Module
```

让我们编译和运行上面的程序，这将产生以下结果：

```
7.33333333333333
```

lambda类型推断

在C#中，类型推断方便地用于各种情况，而且也没有明确指定的类型。但是如果一个lambda表达式，类型推断将工作必须满足在已指定每种类型为编译器。让我们考虑下面的例子。

```
delegate int Transformer (int i);
```

这里，编译器采用的类型推理时的事实，x是一个整数，这是通过检查所述变压器的参数类型进行绘制。

Lambda表达式变量的作用域

有而这样的lambda表达式内发起变量的lambda表达式，使用变量的作用域并不意味着是可见的外部方法有一些规则。还有一个规则，一个捕获变量不是被垃圾回收，除非委托引用相同变得符合垃圾收集的行为。此外，还有禁止lambda表达式中的return语句导致返回封闭方法的规则。

这里有一个例子，以证明lambda表达式变量的作用域。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace lambdaexample
{
    class Program
    {
        delegate bool D();
        delegate bool D2(int i);

        class Test
        {
            D del;
            D2 del2;
            public void TestMethod(int input)
            {
                int j = 0;
                // Initialize the delegates with lambda expressions.
                // Note access to 2 outer variables.
                // del will be invoked within this method.
                del = () => { j = 10; return j > input; };

                // del2 will be invoked after TestMethod goes out of scope.
                del2 = (x) => { return x == j; };

                // Demonstrate value of j:
                // The delegate has not been invoked yet.
                Console.WriteLine("j = {0}", j);    // Invoke the delegate.
                bool boolResult = del();

                Console.WriteLine("j = {0}. b = {1}", j, boolResult);
            }

            static void Main()
            {
                Test test = new Test();
                test.TestMethod(5);

                // Prove that del2 still has a copy of
                // local variable j from TestMethod.
                bool result = test.del2(10);

                Console.WriteLine(result);

                Console.ReadKey();
            }
        }
    }
}
```

```
    }  
  }  
}  
}
```

让我们编译和运行上面的程序，这将产生以下结果：

```
j = 0  
j = 10. b = True  
True
```

表达式树

Lambda表达式中使用表达式树结构广泛。表达式树放弃代码中的数据结构类似于树，其中每个节点本身是一样的方法调用的表达，或者可以是一个二进制运算如 $x < y$ ，下面是lambda表达式的使用用于构造一个表达式树的一个例子。

LAMBDA语句

还有lambda表达式由两个或三个语句，但不仅在构造表达式树中。return语句必须写在lambda语句。

lambda声明的语法

```
(params) => {statements}
```

lambda的声明示例

```
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Linq.Expressions;
```

```
namespace lambdaexample  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            int[] source = new[] { 3, 8, 4, 6, 1, 7, 9, 2, 4, 8 };  
  
            foreach (int i in source.Where(  
                x =>  
                {  
                    if (x <= 3)  
                        return true;  
                    else if (x >= 7)  
                        return true;  
                    return false;  
                }  
            ))  
                Console.WriteLine(i);  
            Console.ReadLine();  
        }  
    }  
}
```

```
}  
}
```

让我们编译和运行上面的程序，这将产生以下结果：

```
3  
8  
1  
7  
9  
2  
8
```

lambda表达式被用作基于方法的LINQ查询参数，并决不允许有一个地方对操作的左侧像是或者就像匿名方法。虽然，Lambda表达式何其相似匿名方法，这些根本不是限制被用来作为唯一表示。

使用lambda表达式要点要记住

- lambda表达式可以返回一个值，并可以带有参数。
- 参数可以用不同的方式使用lambda表达式无数的定义。
- 如果在一个lambda表达式单独的语句，就没有必要花括号，而如果有多条语句，大括号以及返回值都是写必不可少的。
- 随着lambda表达式，可以通过被称为闭合的特征来访问变量lambda表达式块的存在之外。利用闭合的应谨慎，以避免任何问题。
- 这是不执行任何lambda表达式中的任何不安全的代码。
- lambda表达式并不意味着使用操作符的左侧。

易百教程移动端：请扫描本页面底部(右侧)二维码并关注微信公众号，回复："教程" 选择相关教程阅读或直接访问：<http://m.yiibai.com>。

上一篇：[LINQ实体](#) 下一篇：[LINQ ASP.Net](#)

加QQ群啦，易百教程官方技术学习群

所有其它免费技术学习QQ群，请点击：<http://www.yiibai.com/siteinfo/qq-group.html>

QQ群名称	群号	人数/免费/等级	群介绍
JAVA技术	227270512	2000人/否/LV5	Java基础，JSP (Servlet)，JAVA框架，Java

			高并发架构，Maven等等
MySQL/SQL	418407075	2000人/否/LV5	SQL基础，MySQL基础，MySQL存储过程，视图，触发器等等
大数据开发	655154550	2000人/否/LV5	Spark，zookeeper，kafka，CDH，hive，fulme，hbase等Hadoop云计算生态圈技术
Python技术	287904175	2000人/否/LV5	Python编程，Python Web，Python大数据，Python爬虫，自然语言处理等
测试工程师(新群)	415553199	1000人/是/LV1	软件测试，硬件测试，测试平台开发，黑白盒测试，Labview等技术
前端技术群(新群)	410430016	1000人/是/LV1	CSS,HTML/HTML5,JS/JQuery/VueJS/AngularJS等技术
Linux技术	479429477	2000人/是/LV5	Redhat/Centos,Ubuntu,Shell运维监控

			等技术
PHP开发者	460153241	2000人/是/LV5	PHP基础,PHP高级,网站优化/架构,JS,HTML, JQuery等Web技术
人工智能	456236082	2000人/是/LV5	人工智能,深度学习,算法等技术
Oracle数据库	175248146	2000人/是/LV5	SQL基础, Oracle基础, Oracle存储过程, 视图, 触发器等
Android开发	159629185	2000人/是/LV1	Android开发, Android Studio, Kotlin, Dagger等技术
微软技术	579821706	2000人/是/LV1	C#, ASP.Net, VB.Net, ADO.Net, SQL Server, VBA, Excel等技术
数据分析师	397883996	2000人/是/LV1	R语言, Matlab语言, Pandas等技术

所有其它免费技术学习QQ群, 请点击: <http://www.yiibai.com/siteinfo/qq-group.html>

0

收藏文章



登录

我的社区

来说两句吧...

表情删除后不可恢复，是否删除

取消

确定

图片正在上传，请稍后...

[取消上传](#)

评论内容为空！

还没有评论，快来抢沙发吧！

该评论已关闭!

[易百教程正在使用畅言](#)

去社区看看吧

去热评看看吧



VB.Net反向引用构造 - VB.Net教程™

VB.Net反向引用构造 - VB.Net教程™

- [VB.Net反向引用构造 - VB.Net教程™](#)



Spring JavaConfig实例 - Spring教程™

Spring JavaConfig实例 - Spring教程™

- [Spring JavaConfig实例 - Spring教程™](#)



ASP.Net MVC Web API - ASP.Net MVC教程™

ASP.Net MVC Web API - ASP.Net MVC教程™

- [ASP.Net MVC Web API - ASP.Net MVC教程™](#)



VBA输入框 (InputBox) - VBA教程™

VBA输入框 (InputBox) - VBA教程™

- [VBA输入框 \(InputBox \) - VBA教程™](#)



VB.Net反向引用构造 - VB.Net教程™

VB.Net反向引用构造 - VB.Net教程™




Spring JavaConfig实例 - Spring教程™

Spring JavaConfig实例 - Spring教程™

 [ASP.Net MVC Web API - ASP.Net MVC教程™](#)

ASP.Net MVC Web API - ASP.Net MVC教程™

 [VBA输入框 \(InputBox \) - VBA教程™](#)

VBA输入框 (InputBox) - VBA教程™

热评话题

- [VBA输入框 \(InputBox \) - VBA教程™](#)
- [Python连接MongoDB操作 - MongoDB教程™](#)
- [jstl 标签 - JSP教程™](#)
- [Javascript Array.join\(\)方法 - Javascript教程™](#)
- [Git基础概念 - Git教程™](#)
- [R语言泊松回归 - R语言教程™](#)
- [ASP.Net MVC NuGet包管理 - ASP.Net MVC教程™](#)

推荐/最新教程

- [VueJS教程](#)
- [ASP.Net MVC教程](#)
- [.NET Core教程](#)
- [ASP.Net教程](#)
- [MariaDB教程](#)
- [ADO.Net教程](#)
- [VB.Net教程](#)
- [Pandas教程](#)

最新更新

- [Access创建关系](#)
- [Access关联数据](#)
- [Access备用条件](#)
- [Access参数查询](#)
- [Access创建查询](#)
- [Access操作查询](#)
- [Access查询标准\(条件\)](#)
- [Access查询数据](#)

站点信息

- [意见反馈](#)
- [免责声明](#)

· [关于我们](#)

· [关于捐助](#)

· [所有实例代码下载](#)

易百教程官方QQ群

JAVA技术：227270512

MySQL数据库：418407075

Python技术：287904175

大数据开发：655154550

Linux技术：479429477

PHP/Web技术：460153241

关注微信公众号



Copyright © 2012-2017 [易百教程](#) [yiai.com](#) All Rights Reserved. 备案号：琼ICP备13001417号-3 联系我们：769728683@qq.com [站长统计](#)