表13.1 Lambda表达式的注意事项和例子

|  |  |  |
| --- | --- | --- |
| **注意事项** | **例子** | |
| 从C# 9.0开始，Lambda表达式支持弃元参数。（为了保持向后兼容，如果仅一个参数，那么弃元会导致一个变量声明。） | Action<int, int> x = (\_, \_) =>  Console.WriteLine("这是一个测试。"); | |
| 从C# 10.0开始，Lambda表达式具有一个确定性的类型，所以它们可以和匿名类型声明一起使用。 | | // 从C# 10开始，可以将Lambda表达式  // 赋给一个隐式类型的局部变量。  var v = (int x) => x; | |
| 从C# 10.0开始，lambda表达式可以具有返回类型，甚至可以是void类型。这在无法推断委托类型的情况下特别有用，例如在可能返回null时。 | Action action = void () => { };  var func = short? (long number) =>  number <= short.MaxValue ?  (short)number : null; | |
| 不存在能直接从Lambda表达式中访问的成员，就连object定义的那些方法也不存在。 | // 错误：操作符'.'无法应用于  // 'lambda表达式'类型的操作数  string s = ((int x) => x).ToString(); | |
| 由于Lambda表达式没有类型，所以不能出现在is操作符的左侧 | // 错误：'is'或'as'操作符的第一个操作数  // 不能是Lambda表达式、匿名方法或方法组  bool b = ((int x) => x)  is Func<int, int>; | |
| Lambda表达式只能转换成兼容委托类型。在例子中，返回int的Lambda不能转换成代表“返回bool的方法”的委托类型 | // 错误：Lambda表达式不兼容于  // Func<int, bool>类型  Func<int, bool> f = (int x) => x; | |
| C#不允许在Lambda表达式内部使用跳转语句（break，goto，continue）跳转到Lambda表达式外部；反之亦然。在例子中，Lambda中的break语句试图跳转到Lambda外部的switch语句末尾 | // 错误：控制不能离开匿名方法体  // 或Lambda表达式体  // Func<int, bool>类型  string[] args;  Func<string> f;  switch (args[0])  {  case "/File":  f = () =>  {  if (!File.Exists(args[1]))  break;  return args[1];  };  // ...  } | |
| 对于由Lambda表达式引入的参数和局部变量，其作用域仅限于Lambda主体 | // 错误：当前上下文中不存在  // 名称'first'  Func<int, int, bool> expression =  (first, second) => first > second;  first++; | |
| 编译器的确定性赋值分析机制在Lambda表达式内部检测不到对“外部”局部变量进行初始化的情况 | int number;  Func<string, bool> f =  text => int.TryParse(text, out number);  if (f("1"))  {  // 错误: 使用了未赋值的局部变量  System.Console.Write(number);  } | |