Expresiones lambda

Use una expresión lambda para crear una función anónima. Use el operador de declaración lambda => para separar la lista de parámetros de la lamba de su cuerpo. Una expresión lambda puede tener cualquiera de las dos formas siguientes:

Una lambda de expresión que tiene una expresión como cuerpo:

(input-parameters) => expression

Para crear una expresión lambda, especifique los parámetros de entrada (si existen) a la izquierda del operador lambda y una expresión o bloque de instrucciones en el otro lado.

Toda expresión lambda se puede convertir en un tipo delegado. El tipo delegado al que se puede convertir una expresión lambda se define según los tipos de sus parámetros y el valor devuelto. Si una expresión se puede convertir en uno de los tipos delegados Func. Por ejemplo, una expresión lambda que tiene un parámetro y devuelve un valor se puede convertir en un delegado Func<T,TResult>.

En el ejemplo siguiente, la expresión lambda x => x * x, de la función cuadrado, que especifica un parámetro denominado x y devuelve el valor de x al cuadrado, se asigna a una variable de un tipo delegado:

```
static Func<int, int> cuadrado = x => x * x;
static Func<int, bool> EsTres = b => b == 3;
static Func<Empleado, bool> EsJorge = empleado => empleado.nombre == "Jorge";
1 referencia
public static void Presentacion()
{
    Console.WriteLine(cuadrado(3));
    Console.WriteLine(EsTres(2));
    Console.WriteLine(EsTres(3));
    Empleado empleado1 = new Empleado() { idEstado=1, nombre ="Juan"};
    Empleado empleado2 = new Empleado() { idEstado = 2, nombre = "Jorge" };
    Console.WriteLine(EsJorge(empleado1));
    Console.WriteLine(EsJorge(empleado2));
}
```

```
9
False
True
False
True
```

Las expresiones lambda también se pueden convertir en los tipos de árbol de expresión, como se muestra en los ejemplos siguientes:

```
Expression<Func<int, int>> e = x => x * x;
Console.WriteLine(e);
```

```
x => (x * x)
```

Puede usar expresiones lambda en cualquier código que requiera instancias de tipos delegados o de árboles de expresión, por ejemplo, como un argumento del método Task.Run(Action) para pasar el código que se debe ejecutar en segundo plano. También puede usar expresiones lambda al escribir LINQ en C#, como se muestra en el ejemplo siguiente:

```
int[] entero = { 2, 3, 4, 5 };
var enteroCuadrado = entero.Select(x => x * x);
foreach (int num in enteroCuadrado)
{
    Console.WriteLine(num);
}
```

```
4
9
16
25
```

Cuando se usa la sintaxis de método para llamar al método Enumerable.Select en la clase System.Linq.Enumerable, por ejemplo en LINQ to Objects y en LINQ to XML, el parámetro es un tipo delegado System.Func<T,TResult>. Cuando se llama al método Queryable.Select en la clase System.Linq.Queryable, por ejemplo en LINQ to SQL, el tipo de parámetro es un tipo de árbol de expresión Expression<Func<TSource,TResult>>. En ambos casos, se puede usar la misma expresión lambda para especificar el valor del parámetro. Esto hace que las dos llamadas Select tengan un aspecto similar aunque, de hecho, el tipo de objetos creados a partir las lambdas es distinto.

Lambdas de expresión

Una expresión lambda con una expresión en el lado derecho del operador => se denomina lambda de expresión. Una expresión lambda devuelve el resultado de evaluar la condición y tiene la siguiente forma:

(input-parameters) => expression

El cuerpo de una expresión lambda puede constar de una llamada al método.