Lambda Ausdruck

Ingo Köster

Diplom-Informatiker (FH)

Was ist ein Lambda-Ausdruck?

› Eine anonyme Methode

› Kann Ausdrücke und Anweisungen enthalten

› Kann für Erstellung von Delegates verwendet werden

Lambda-Ausdrücke

› Anonymer Delegate

```
calculate = delegate(double x, double y)
{
  return x + y;
};
```

› Lambda Ausdruck

```
calculate = (double x, double y) => { return x + y; };
```

(parameter) => {Funktionsblock}

› Lambda-Ausdrücke verwenden den Operator **=>**

› Links vom Operator werden die Eingabeparameter angegeben

› Rechts vom Operator der Ausdruck oder ein Anweisungsblock

› Das Schlüsselwort `delegate` wird nicht mehr verwendet

Lambda-Ausdrücke -> Kürzer

› Häufig treten Lambda-Ausdrücke mit nur einer `return` Anweisung auf

› Dann kann die `return`-Anweisung wegfallen
  › Gleichzeitig auch die geschweiften Klammern

› Anstatt:

```
calculate = (double x, double y) => { return x + y; };
```

› Kürzer:

```
calculate = (double x, double y) => x + y;
```

Lambda-Ausdrücke -> Noch Kürzer

› `calculate = (x, y) => x + y;`

› Angabe der Parametertypen wurde entfernt

› Es handelt sich um implizit typisierte Parameter
  › Compiler leitet die Parametertypen von der Delegaten-Deklaration ab
  › `delegate double CalculateHandler(double x, double y);`

Lambda-Ausdrücke -> Noch Kürzer

- Liegt nur ein Parameter vor, können die runden Klammern weggelassen werden
  - `x => x + x;`

- Bei einem Lambda-Ausdruck mit leerer Parameterliste müssen die runden Klammen angegeben werden
  - `() => Console.WriteLine("Hallo Welt");`

Von lang zu kurz

```
delegate double CalculateHandler(double x, double y);
CalculateHandler calculate;
```

› Lang:

```
calculate = delegate(double x, double y)
            {
               return x + y;
            };
```

› Kurz:

```
calculate = (x, y) => x + y;
```

Lambda Ausdruck ohne Delegaten deklarieren

› Mittels generischem Delegate Func

```
public delegate TResult Func<in T, out TResult>(T arg)
```

› Beispiel:

```
Func<double,double,double> addiere = (x, y) => x + y;
```

- › 1. und 2. double: Übergabeparameter (x und y)
- › 3. double: Rückgabewert (immer am Ende)

› Aufruf :

```
double ergebnis = addiere(3, 6);
```

Lambda Ausdruck ohne Delegaten deklarieren

- Func bietet bis zu 16 Parameter an
- `public delegate TResult Func<in T1, in T2, in T3, in T4, in T5, in T6, in T7, in T8, in T9, in T10, in T11, in T12, in T13, in T14, in T15, in T16, out TResult> (T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7, T8 arg8, T9 arg9, T10 arg10, T11 arg11, T12 arg12, T13 arg13, T14 arg14, T15 arg15, T16 arg16)`

Lambda Ausdruck ohne Delegaten deklarieren

- Delegat Action wie Func, jedoch ohne Rückgabeparameter
  - `public delegate void Action<in T>(T obj)`

- Beispiel:

```
Action hallo = () => Console.WriteLine("Hallo Welt");
```

- Weiteres Beispiel (generisch):

```
Action<string> halloString = (name) => Console.WriteLine("Hallo " +
  name);
```

- Aufruf:

```
halloString("Welt");
```

Sichtbarkeitsbereich von Variablen

› Lambda Ausdrücke (und anonyme Methoden) werden häufig innerhalb von Methoden deklariert

› Innerhalb dieser Lambda Ausdrücke stehen die Variablen, welche in der umschließenden Methode erstellt wurden, zur Verfügung

› Beispiel

```
int x = 42;
Action ausgabe = () => Console.WriteLine(x);
```

Beispiele Lambda Ausdrücke

Beispiel Lambda Ausdruck

› Klasse Person mit Namen und Alter

```
class Person {
  public string name;
  public int alter;
}
```

› Sortieren mittels Sort und Lambda Ausdruck

```
› Array.Sort(personenArray,
(p1, p2) => p1.alter.CompareTo(p2.alter));
```

Beispiel Lambda Ausdruck

› Liste mit Zahlenwerten zwischen 0 und 20 füllen

```
List<int> zahlen = new List<int>();
Random random = new Random();
for (int i = 0; i < 10; i++)
{
  zahlen.Add(random.Next(21));
}
```

› Entferne alle Zahlen kleiner 10 aus der Liste (Predicate Delegate)

› `zahlen.RemoveAll(x => x < 10);`

Predicate<T>-Delegat

› Stellt eine Methode dar, die einen Satz von Kriterien definiert und bestimmt, ob das angegebene Objekt jene Kriterien erfüllt

› Rückgabewert ist immer bool

› Ist `true` , wenn das Objekt die innerhalb des Delegaten definierten Kriterien erfüllt, andernfalls `false`

Beispiel – Mehrzeiliger Lambda Ausdruck

› Ein Lambda Ausdruck kann mehrere Zeilen verwenden

› Dann müssen die geschweiften Klammern verwendet werden

› Zudem muss eine `return`-Anweisung verwendet werden

› Beispiel:

```
Func<int, int, double> dividiere = (x, y) =>
{
 Console.WriteLine("{0} / {1}", x, y);
 return (double)x / y;
};
```