Erweiterungsmethoden - Extension Methods

Ingo Köster

Diplom-Informatiker (FH)

Erweiterungsmethoden

- Mit Erweiterungsmethoden können vorhandenen Typen Methoden hinzugefügt werden, ohne einen neuen abgeleiteten Typ zu erstellen

- Hilfreich bei Klassen, von denen nicht abgeleitet werden kann
  - `sealed` Klassen

- Erweiterungsmethoden sind eine besondere Art von statischen Methoden

Beispiel – String mit großem Anfangsbuchstaben

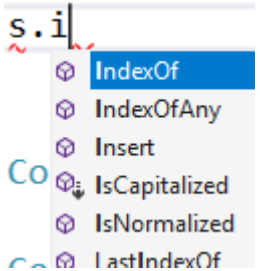
```
static class StringHelper
{
 public static bool IsCapitalized(this string s)
 {
  if (string.IsNullOrEmpty(s))
  {
   return false;
  }

  return char.IsUpper(s[0]);
 }
}
```

Beispiel – String mit großem Anfangsbuchstaben

› Aufrufbeispiele:

```
string s = "Hallo";
s.IsCapitalized()
"Hallo".IsCapitalized()
```



› Erweiterungsmethoden werden trotz der `static` Definition wie Instanzmethoden aufgerufen

› Der erste Übergabeparameter einer Erweiterungsmethode muss das Schlüsselwort `this` vor dem Typ verwenden

  › Dieser Typ wird um die Methode erweitert

Erweiterungsmethoden

- Erweiterungsmethoden werden immer in statischen Klassen definiert
- Erweiterungsmethoden sind öffentliche und statische Methoden
- Wird eine Klasse um Erweiterungsmethoden erweitert und anschließend wird von dieser geerbt, enthalten auch die Kindklassen diese Methoden

Erweiterungsmethoden nach dem Kompilieren

- Aufruf einer Erweiterungsmethode wird vom Compiler in einen normalen statischen Methodenaufruf umgewandelt

- Aus
  - `Console.WriteLine("Hallo".IsCapitalized());`
- wird
  - `Console.WriteLine(StringHelper.IsCapitalized("Hallo"));`

Mehrdeutigkeit

› Instanz-Methoden haben immer Vorrang vor Erweiterungsmethoden

```
class MyClass
{
 public void EineMethode(int x) {
  Console.Write("MyClass.EineMethode: {0}", x);
 }
}

static class Extensions
{
 public static void EineMethode(this MyClass my, int x) {
  Console.Write("Extensions.EineMethode: {0}", x);
 }
}
```

Erweiterungsmethoden für Interfaces

Erweiterungsmethoden für Interfaces

```
// Schnittstelle
public interface IMyInterface {
 void MethodeA();
}

// Erweiterungsmethode für die Schnittstelle
static class Extensionmethods {
  public static void MethodeB(this IMyInterface my)
  {            Console.WriteLine("Extensionmethods.MethodeB
              (this IMyInterface myInterface)");
  }
}
```

Erweiterungsmethoden für Interfaces

› Jede Klasse, welche das Interface `IMyInterface` implementiert, wird um die Methode `MethodeB` erweitert

```
class MyClass : IMyInterface
{
 public void MethodeA() {
Console.WriteLine("MyClass.MethodeA()");
 }
}

MyClass m = new MyClass();
m.MethodeA();
m.MethodeB();
```

Erweiterungsmethoden für Interfaces

› Weiteres Beispiel: Alle Typen, die `IEnumerable` implementieren um eine Methode `First` erweitern

```
static class Extensionmethods {
 public static T First<T>(this IEnumerable<T> sequence) {
  foreach (T element in sequence) {
   return element;
  }

  throw new InvalidOperationException("Keine Elemente");
 }
}
```

Erweiterungsmethoden für Interfaces

› Aufruf-Beispiele:

```
string s = "Hallo";
Console.WriteLine(s.First());   // H

int[] zahlen = { 42, 23, 99, 14 };
Console.WriteLine(zahlen.First());  // 42

List<Person> personen = new List<Person>() { new Person { Name =
  "Karl" }, new Person { Name = "Hans" }, new Person { Name =
  "Franz" } };
Console.WriteLine(personen.First()); // Karl
```

Hinweise

› Auch Wertetypen wie `int`, `float`, etc. und Strukturen (`struct`) können mittels Erweiterungsmethoden erweitert werden

```
static class IntExtension
{
  public static bool IsOdd(this int value)
  {
    return ((value % 2) == 1);
  }
}
```

› Statische Klassen wie z.B. `Math` oder `Console` können nicht durch Erweiterungsmethoden erweitert werden