Richard Kaiser

www.rkaiser.de

C# Aufbaukurs

mit

Microsoft Visual Studio 2015/2017

Inhalt

1 Weiterführende Möglichkeiten der Entwicklungsumgebung

Die in diesem Abschnitt vorgestellten Möglichkeiten sind für kleinere Projekte wie die Übungen in diesem Buch nicht unbedingt notwendig. Anfänger können diese zunächst übergehen. Für größere Projekte können sie aber doch sehr nützlich sein.

1.1 Der Quelltexteditor

Der Quelltexteditor (kurz: **Editor**) ist das Werkzeug, mit dem die Quelltexte geschrieben werden. Er ist in die Entwicklungsumgebung integriert und kann auf verschiedene Arten aufgerufen werden, wie z.B.

- durch Anklicken seines Registers 

- über *Ansicht*|*Code* oder *F7*
- von einem Formular aus über das Kontextmenü „Code anzeigen“
- durch einen Doppelklick auf die rechte Spalte eines Ereignisses im Eigenschaftenfenster. Der Cursor befindet sich dann in der Methode, die zu dem angeklickten Ereignis gehört.
- durch einen Doppelklick auf ein Steuerelement in einem Formular. Der Cursor befindet sich dann in einer Methode, die zu einem bestimmten Ereignis des Steuerelements gehört.
  Auf diese Weise kann man auch eine Methode finden, ohne sie lange im Editor suchen zu müssen.

1.1.1 Tastenkombinationen

Der Editor enthält über Tastenkombinationen zahlreiche Funktionen, mit denen sich nahezu alle Aufgaben effektiv durchführen lassen, die beim Schreiben von Programmen auftreten.

Die nächste Tabelle enthält Funktionen, die vor allem beim Programmieren nützlich sind, und die man in einer allgemeinen Textverarbeitung nur selten findet. Die meisten dieser Optionen werden auch unter *BEARBEITEN*|*Erweitert* sowie auf der *Text Editor* Symbolleiste (unter *ANSICHT*|*Symbolleisten*) angezeigt:

Tastenkürzel	Aktion oder Befehl
F5 bzw. 	kompilieren und starten, wie *Debuggen*|*Debugging Starten*
Umschalt+F5	Laufendes Programm beenden, wie *Debuggen*|*Debugging beenden*. Damit können auch Programme beendet werden, die mit  nicht beendet werden können. Versuchen Sie immer zuerst diese Option wenn Sie meinen, Sie müssten Visual Studio mit dem Windows Task Manager beenden.
F6	kompilieren, aber nicht starten
F7	wie *Ansicht*|*Code*
Alt|*Enter*	wie *Ansicht*|*Eigenschaftenfenster*
Umschalt+F7	wie *Ansicht*|*Designer*

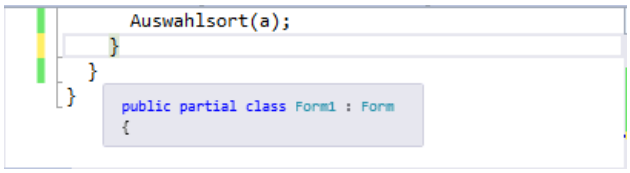
Tastenkürzel	Aktion oder Befehl
F1	kontextsensitive Hilfe
Strg +´ (´ ist das Zeichen links von der Rücktaste) *Umschalt+Strg* +´	setzt den Cursor vor die zugehörige Klammer, wenn er vor einer Klammer (z.B. (), {}, [ ] oder <>) steht markiert den Bereich zwischen den Klammern außerdem noch
Strg+M+M bzw. unter *Bearbeiten|Gliedern*	ganze Funktionen, Klassen usw. auf- oder zuklappen

Alt+Maus bewegen bzw. *Alt+Umschalt*+Pfeiltaste (←, →, ↑ oder ↓)	zum Markieren von Spalten , z.B.
	`/// <summary>`
	`/// Clean up any`
	`/// </summary>`
	`/// <param name=`

Strg+K+C oder *Strg+K+U* bzw.  oder 	einen markierten Block auskommentieren bzw. die Auskommentierung entfernen

Eine ausführliche Liste der Tastenkombinationen findet man in der MSDN-Dokumentation mit dem Suchbegriff „Visual Studio Tastenkombinationen“.

Bei größeren Programmen ist oft nicht sofort klar, ob eine schließende geschweifte Klammer eine Klasse, einen Namensbereich, eine Funktion oder eine Anweisung (*if*, *while* usw.) beendet. Seit Visual Studio 2015 wird der Anfang zu einer solchen Klammer im Editor eingeblendet, wenn man mit dem Mauszeiger darüber fährt:



Weitere Möglichkeiten des Editors, die gelegentlich nützlich sind:

- Die Schriftgröße kann man mit *Strg*+Mausrad vergrößern und verkleinern.
- Wenn man verschiedene Teile eines Textes in verschiedenen Fenstern anschauen (und z.B. vergleichen will), kann man die aktuelle Datei in verschiedenen Fenstern öffnen:

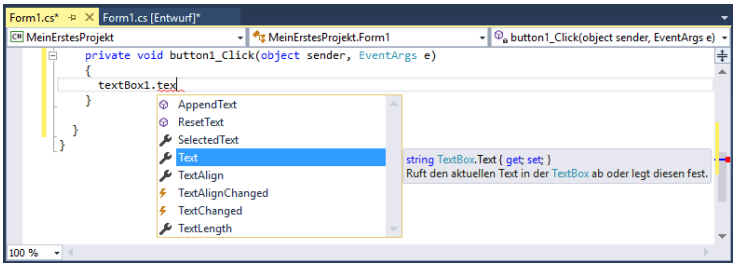
1.1.2 Intellisense

Die folgenden Programmierhilfen beruhen auf einer Analyse des aktuellen Programms. Sie werden zusammen mit einigen weiteren (siehe *Bearbeiten|IntelliSense*) unter dem Oberbegriff **IntelliSense** zusammengefasst:

- Der Editor bietet fast jederzeit eine Auswahl von **zulässigen Sprachelementen** an. Wenn ein Formular z.B. eine *textBox1* enthält, wird in einer ButtonClick-Funktion des Formulars nach dem Eintippen von „tex“ textBox1 angeboten. Außerhalb des Formulars (z.B. am Ende von Form1.cs) ist der Name textBox1 nicht bekannt und wird auch nicht angeboten.
- **Elemente anzeigen und auswählen**: Nachdem man den Namen eines Steuerelements (genauer: einer Klasseninstanz, eines Namensbereichs usw.) und einen Punkt „.“ eingetippt hat, wird eine Liste mit allen Elementen der Klasse oder des Namensbereichs angezeigt. Aus dieser Liste kann man mit der Enter-Taste ein Element auswählen.
- **Parameter Info**: Zeigt nach dem Eintippen eines Funktionsnamens und einer öffnenden Klammer die Parameter der Funktion an
- **Quick Info**: Wenn man mit der Maus über einen Namen für ein zuvor deklariertes Symbol fährt (bzw. mit *Strg+K+I*), wird die Deklaration angezeigt. Bei einer vordefinierten Funktion wird außerdem eine Beschreibung angezeigt.

Beispiel: Wenn das Formular eine TextBox *textBox1* enthält, wird nach dem Eintippen von „textBox1.“ eine Liste mit allen Elementen von *textBox1* angezeigt.

Tippt man weitere Buchstaben ein, werden nur die Elemente angezeigt, die diese Zeichenfolge enthalten. In einem weiteren Fenster werden außerdem der Datentyp und eine kurze Beschreibung der Eigenschaft angezeigt.



Tippt man weitere Buchstaben ein, werden nur die Elemente mit diesen Anfangsbuchstaben angezeigt. In einem weiteren Fenster werden außerdem der Datentyp und eine kurze Beschreibung der Eigenschaft angezeigt.

Unterbricht man die Eingabe einer Zeichenfolge (z.B. weil man die Zeile verlässt und anschließend wieder zurückkehrt), bietet Intellisense keine Vorschläge an. Mit *Strg+Leerzeichen* werden dann wieder Vorschläge angezeigt.

Falls der Editor einen **Syntaxfehler** entdeckt, zeigt er das mit einer rot unterstrichenen Wellenlinie an (wie bei Rechtschreibfehlern in Word).

1.1.3 Die Formatierung des Quelltexts

Wenn der Editor erkennt, dass die Eingabe einer Anweisung, eines Blocks usw. abgeschlossen wurde (z.B. nach einer Anweisung beim Eintippen des Semikolons, bei einer Funktion nach dem Eintippen der schließenden geschweiften Klammer „}“ usw.) „verschönert“ der Editor das Layout. Tippt man z.B.

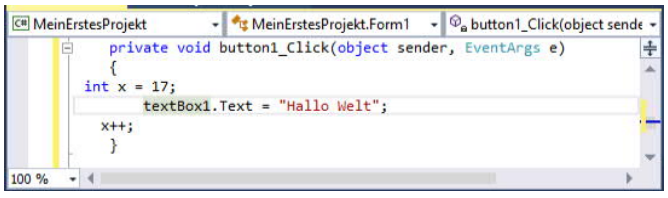
```
int x=17
```

macht der Editor daraus mit dem Eintippen des „;“

```
int x = 17;
```

Diese automatische Formatierung erfolgt aber nur bei beim Eintippen eines abschließenden Symbols. Entfernt man die Leerzeichen anschließend wieder, wird der Text nicht erneut formatiert.

Dann kann man den Text mit *Bearbeiten|Erweitert|Dokument formatieren* formatieren lassen. Damit wird z.B. aus



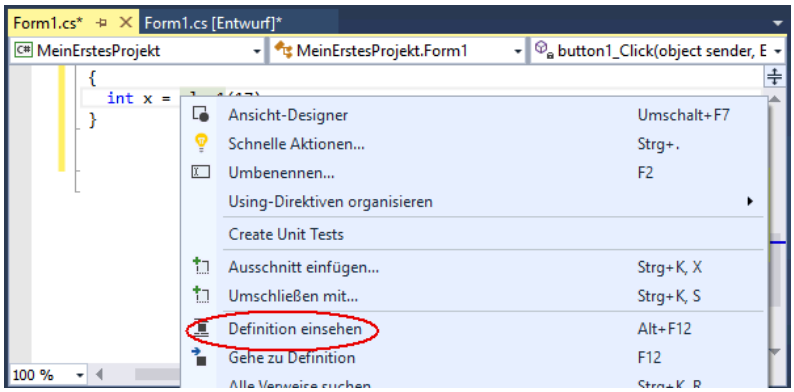
dieser Text

bei dem die Anweisungen einer Funktion in derselben Spalte beginnen.

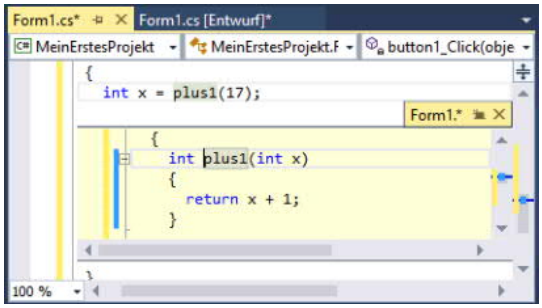
Unter *Extras|Optionen|Text-Editor|C#|Formatierung* gibt es zahlreiche Möglichkeiten, das so erzeugte Layout zu gestalten.

1.1.4 Definitionen einsehen

Beim Aufruf einer Funktion oder der Verwendung einer Variablen besteht immer wieder die Notwendigkeit, die Definition einer Funktion bzw. Variablen anzuschauen. Das ist mit der Option *Definition einsehen* aus dem Kontextmenü zu einer Anweisung besonders einfach, da sie das Suchen und Blättern im Quelltext erspart und man nicht einmal die aktuelle Position im Editor verlassen muss:



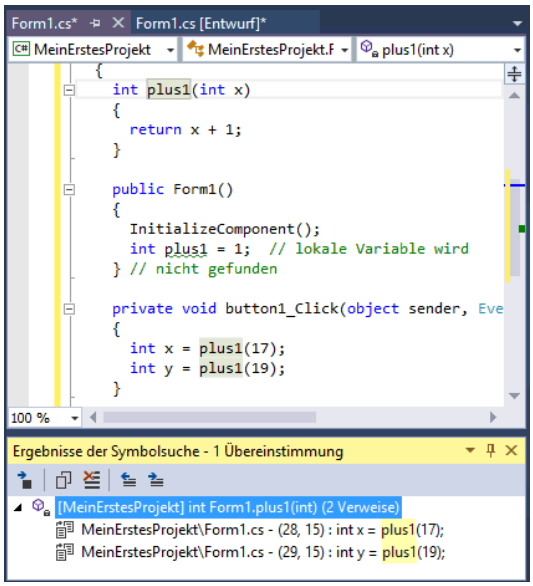
Damit wird ein Fenster mit dem Quelltext eingeblendet, indem man sogar Änderungen vornehmen kann.



Falls man trotzdem noch zur Definition oder Deklaration gehen will, ist das *Gehe zu Definition* bzw. *Gehe zu Deklaration* aus dem Kontextmenü möglich.

1.1.5 Symbole suchen

Alle Aufrufe einer Funktion bzw. alle Stellen, an denen eine Variable verwendet wird, kann man mit der Option „Alle Verweise suchen“ aus dem Kontextmenü im Editor anzeigen lassen:

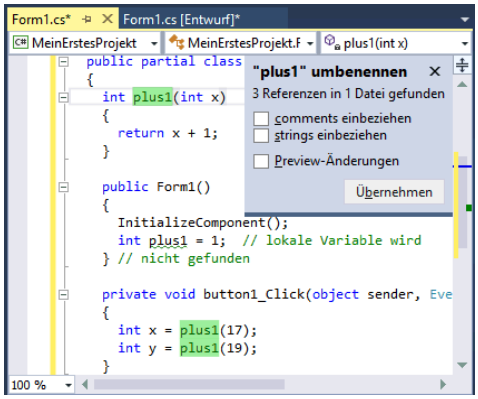


Die Treffer sind hiermit viel präziser als mit *Bearbeiten|Suchen* von Abschnitt 1.1.7.

1.1.6 Namen umbenennen

Namen, die in C# definiert wurden (Namen von Variablen, Funktionen, Klassen usw.) können mit ***Umgestalten|-Umbenennen*** im gesamten Quelltext geändert werden. Da die meisten Zeichenfolgen, die man in einem Programm ersetzen will, solche Namen sind, ist die mit *Bearbeiten|Suchen und Ersetzen* (siehe Abschnitt 1.1.7) verfügbare Option nur noch selten notwendig.

Klickt man im Editor mit der rechten Maustaste den Namen einer Variablen, Methode usw. an, wird im Kontextmenü die Option ***Umgestalten|Umbenennen*** angeboten, mit der man diese Variable, Methode usw. im gesamten Quelltext umbenennen kann. Wenn der Name anders verwendet wird (z.B. für eine andere Variable mit demselben Namen), wird er nicht geändert. Ändert man hier den Namen, wird die Änderung an allen Stellen angezeigt.



Vor der Durchführung der Änderungen können diese in einer Vorschau angezeigt werden.

Diese Option steht auch nach dem Ändern eines Namens im Editor zur Verfügung. Nach der Änderung wird das geänderte Symbol mit einem gestrichelten Rechteck umrandet und das Glühbirnen-Symbol angezeigt: