**Visual C#: Arbeiten mit INI-Dateien**

19.09.2015, Rafael Nockmann  
Kategorie: [Tutorials](https://nocksoft.de/kategorie/tutorials/), [Visual C#](https://nocksoft.de/kategorie/tutorials/visual-c-sharp/)

Auch wenn [INI-Dateien](https://de.wikipedia.org/wiki/Initialisierungsdatei) etwas aus der Mode gekommen sind, ist es durchaus legitim mit Ihnen zu arbeiten und Informationen darin zu speichern. Um dies ohne größere Probleme zu tun, habe ich eine Klasse in C# geschrieben, deren Anwendung ich in diesem Artikel erklären möchte.

Inhaltsverzeichnis

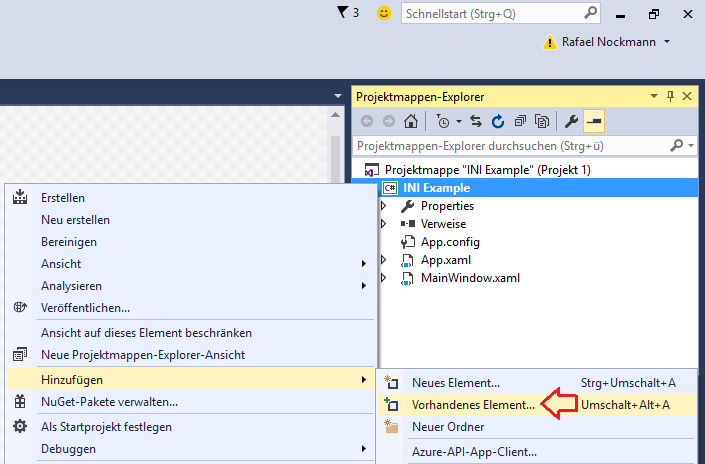
* [1 Klasse herunterladen und einbinden](https://nocksoft.de/tutorials/visual-c-sharp-arbeiten-mit-ini-dateien/#Klasseherunterladenundeinbinden)
* [2 Namespace einbinden](https://nocksoft.de/tutorials/visual-c-sharp-arbeiten-mit-ini-dateien/#Namespaceeinbinden)
* [3 Objekt der Klasse 'INIFile' erzeugen](https://nocksoft.de/tutorials/visual-c-sharp-arbeiten-mit-ini-dateien/#ObjektderKlasseINIFileerzeugen)
* [4 Methode 'GetValue()'](https://nocksoft.de/tutorials/visual-c-sharp-arbeiten-mit-ini-dateien/#MethodeGetValue)
* [5 Methode 'SetValue()'](https://nocksoft.de/tutorials/visual-c-sharp-arbeiten-mit-ini-dateien/#MethodeSetValue)
* [6 Methode 'GetSection()'](https://nocksoft.de/tutorials/visual-c-sharp-arbeiten-mit-ini-dateien/#MethodeGetSection)
* [7 Fehlerbehandlung](https://nocksoft.de/tutorials/visual-c-sharp-arbeiten-mit-ini-dateien/#Fehlerbehandlung)
* [8 Kleines Beispielprogramm](https://nocksoft.de/tutorials/visual-c-sharp-arbeiten-mit-ini-dateien/#KleinesBeispielprogramm)

Hinweis:  
Dieser Artikel basiert auf Version 1.0.3 vom 09.09.2017 der Klasse 'INIFile'.

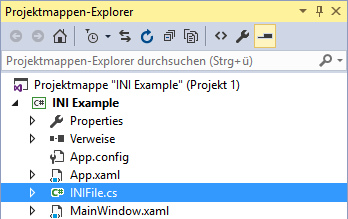
**Klasse herunterladen und einbinden**

Als erstes ist es notwendig die Klasse [herunterzuladen](https://nocksoft.de/downloads/_quellcode/dotnet/inifile.zip). Bitte beachtet, dass die Klasse unter folgender [Lizenz](https://nocksoft.de/downloads/_lizenzen/nocksoft-lizenz-quellcode.txt) von mir veröffentlicht wurde.

Nachdem das heruntergeladene ZIP-Paket entpackt ist, kann die Klasse in einem Visual Studio Projekt eingebunden werden.



Anschließend wird die Klasse 'INIFile' im Projektmappen-Explorer angezeigt:



**Namespace einbinden**

Um mit der Klasse arbeiten zu können, muss in der Klasse, in der wir mit 'INIFile' arbeiten möchten, der Namespace 'Nocksoft.IO.ConfigFiles' eingebunden werden. Dafür ist folgender Code bei den using-Direktiven hinzuzufügen:



|  |  |
| --- | --- |
| 1 | using Nocksoft.IO.ConfigFiles; |

**Objekt der Klasse 'INIFile' erzeugen**

Jetzt kann eine neue Instanz der Klasse 'INIFile' erzeugt werden. Ich nenne das Objekt 'iniFile'. Als Parameter erwartet der Konstruktor der Klasse 'INIFile' die Pfadangabe der INI-Datei, in der wir lesen oder schreiben wollen. Dafür eignet sich an gewünschter Stelle folgender Code:



|  |  |
| --- | --- |
| 1 | INIFile iniFile = new INIFile("settings.ini"); |

Jetzt stehen mehrere Methoden zur Verfügung. Die wichtigsten Methoden sind 'GetValue()' und 'SetValue()'. Denn mit diesen können Werte aus der INI-Datei gelesen bzw. in diese hineingeschrieben werden.

**Methode 'GetValue()'**

Mit 'GetValue()' lässt sich ein Wert auslesen. Dazu muss angegeben werden, in welcher Sektion und zu welchem Schlüssel der Wert gehört. Ein Aufruf sieht dann beispielsweise so aus:



|  |  |
| --- | --- |
| 1 | string value = iniFile.GetValue("section", "key"); |

Als optionalen Parameter kann der Methode noch ein Wahrheitswert übergeben werden. Wird 'true' übergeben, wird der Rückgabewert in Kleinbuchstaben konvertiert. Dadurch ist das Programm unempfindlich gegen Groß- und Kleinschreibung. Der Defaultwert ist an dieser Stelle 'false'.

**Methode 'SetValue()'**

Um einen Wert in die INI-Datei zu schreiben oder zu ändern, eignet sich die Methode 'SetValue()'. Auch dieser Methode sind die Parameter ’section' und 'key' zu übergeben. Als dritten Parameter erwartet die Methode noch einen String, der in die INI-Datei geschrieben werden soll. Wenn die Sektion oder der Schlüssel nicht vorhanden sind, werden diese in der INI-Datei angelegt. Genau wie bei 'GetValue()' kann hier wieder optional ein Wahrheitswert übergeben werden. Wird 'true' übergeben, wird der Wert in Kleinbuchstaben abgespeichert. Der Defaultwert ist an dieser Stelle ebenfalls 'false'. Hier ein Beispiel des Aufrufs:



|  |  |
| --- | --- |
| 1 | iniFile.SetValue("section", "key", "value"); |

**Methode 'GetSection()'**

Die Methode 'GetSection()' gibt den gesamten Inhalt einer Section aus der INI-Datei als Liste zurück. Optional auch mit Kommentaren. Ein Aufruf sieht dann etwa so aus:



|  |  |
| --- | --- |
| 1 | List<string> section = iniFile.GetSection("section"); |

Sollen auch Kommentare aus der INI-Datei mit zurückgegeben werden, ist als zweiter Parameter der Wahrheitswert 'true' zu übergeben.

**Fehlerbehandlung**

Es kann sein, dass die INI-Datei, die übergeben wird, nicht vorhanden ist. Daher ist es ratsam diesen Fehler mit einer Überprüfung oder einem Try-Catch-Block abzufangen. Dies könnte dann beispielsweise so aussehen:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | try  {      INIFile iniFile = new INIFile("settings.ini");      string value = iniFile.GetValue("section", "key");      iniFile.SetValue("section", "key", "value");  }  catch(System.IO.FileNotFoundException)  {      MessageBox.Show("Datei nicht gefunden.");  } |

**Kleines Beispielprogramm**

Im folgenden Beispiel wird als erstes ein Objekt vom Typ 'INIFile' erzeugt. Mit diesem greife ich auf die Methode 'SetValue()' zu. Es wird in der Sektion 'anwender' für den Schlüssel ’name' der Wert geschrieben, den der Anwender in die TextBox eingegeben hat. Im nächsten Schritt wird dieser Wert wieder aus der INI-Datei gelesen, um ihn in einer MessageBox auszugeben. Der Try-Catch-Block fängt den Fehler ab, dass die INI-Datei nicht vorhanden ist.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | using System.Windows;  using Nocksoft.IO.ConfigFiles;  namespace INI\_Example  {      /// <summary>      /// Interaktionslogik für MainWindow.xaml      /// </summary>      public partial class MainWindow : Window      {          public MainWindow()          {              InitializeComponent();          }          private void button\_Click(object sender, RoutedEventArgs e)          {              try              {                  // Instanz der Klasse 'INIFile' erzeugen                  INIFile iniFile = new INIFile("settings.ini");                  // Wert in INI-Datei speichern                  iniFile.SetValue("anwender", "name", textBox.Text);                  // Wert aus INI-Datei auslesen                  string value = iniFile.GetValue("anwender", "name");                  MessageBox.Show(value);                }              catch(System.IO.FileNotFoundException)              {                  MessageBox.Show("Datei nicht gefunden.");              }          }      }  } |

Ich hoffe, die Klasse stößt auf Gefallen und nimmt euch etwas Arbeit ab.