

## C# Dateibearbeitung

---

### Der einfachste Weg, in eine Datei zu schreiben und daraus zu lesen

Der Weg, um in eine Datei zu schreiben oder eine Datei zu lesen, bedurfte bisher immer mehrerer Codezeilen. Eigentlich viel zu viel Aufwand. Das .NET Framework bietet eine Vielzahl an Möglichkeiten um Dateizugriffe zu vereinfachen. So bietet z.B. die Klasse `File` die Möglichkeit mit nur einer einzigen Zeile Code Daten in eine Datei zu schreiben beziehungsweise mit einer Zeile Code Daten einzulesen. Es handelt sich hier um die Methoden *ReadAllBytes*, *ReadAllLines* und *ReadAllText* zum Lesen und *WriteAllBytes*, *WriteAllLines* und *WriteAllText* zum schreiben.

Mit der simplen Anweisung

```
File.WriteAllText(@"C:\MyTextFile.txt", strText);
```

können man bereits den Inhalt der Variablen `strText` in die angegebene Datei schreiben. Existiert die Datei schon, wird sie einfach überschrieben.

## C# Dateibearbeitung - Klasse File

---

Die nicht ableitbare Klasse File stellt nur *statische Methoden* zur Verfügung, die Methoden von *FileInfo* sind Instanzmethoden und werden über eine Objektreferenz aufgerufen. Funktionell sind sich beide Klassen jedoch ähnlich und unterscheiden sich nicht gravierend.

### Methoden der Klasse File

Mit den Klassenmethoden von File lässt sich eine Datei erstellen, kopieren, löschen usw. Man kann auch die Attribute einer Datei lesen oder setzen, und – was auch sehr wichtig ist – man kann eine Datei öffnen.

Um das Lesen und Schreiben von Dateien zu vereinfachen, wurde die Klasse mit dem .NET Framework 2.0 um einige Methoden erweitert. In der folgenden Tabelle sind die wichtigsten Methoden aufgeführt.

## C# Dateibearbeitung - Klasse File

Methodenname	Rückgabetypp	Beschreibung
<i>AppendAllText</i>	void	Öffnet eine Datei, fügt die angegebene Zeichenfolge an die Datei an.
<i>AppendText</i>	StreamWriter	Hängt Text an eine existierende Datei an.
<i>Copy</i>	void	Kopiert eine bestehende Datei an eine andere Speicherlokalität.
<i>Create</i>	FileStream	Erzeugt eine Datei in einem angegebenen Pfad.
<i>CreateText</i>	StreamWriter	Erstellt oder öffnet eine Textdatei.
<i>Delete</i>	void	Löscht eine Datei.
<i>Exists</i>	Boolean	Gibt einen booleschen Wert zurück, der false ist, wenn die angegebene Datei nicht existiert.
<i>GetAttributes</i>	FileAttributes	Liefert das Bitfeld der Dateiattribute.
<i>GetCreationTime</i>	DateTime	Liefert das Erstellungsdatum und die Uhrzeit einer Datei.
<i>GetLastAccessTime</i>	DateTime	Liefert Datum und Uhrzeit des letzten Zugriffs.
<i>GetLastWriteTime</i>	DateTime	Liefert Datum und Uhrzeit des letzten Schreibzugriffs.
<i>Move</i>	void	Verschiebt eine Datei in einen anderen Ordner oder benennt sie um.
<i>Open</i>	FileStream	Öffnet eine Datei.
<i>OpenRead</i>	FileStream	Öffnet eine Datei zum Lesen.
<i>OpenText</i>	StreamReader	Öffnet eine Textdatei zum Lesen.
<i>OpenWrite</i>	FileStream	Öffnet eine Datei zum Schreiben.
<i>ReadAllBytes</i>	byte[]	Öffnet eine Binärdatei, liest den Inhalt der Datei in ein Byte-Array ein.
<i>ReadAllLines</i>	string[]	Öffnet eine Textdatei, liest alle Zeilen der Datei in ein Zeichenfolgen-Array ein.

## C# Dateibearbeitung - Klasse File

<b>Methode</b>	<b>Rückgabetyyp</b>	<b>Beschreibung</b>
<i>ReadAllText</i>	string	Öffnet eine Textdatei, liest alle Zeilen der Datei in eine Zeichenfolge ein und schließt dann die Datei.
<i>SetAttributes</i>	void	Setzt Dateiattribute.
<i>SetCreationTime</i>	void	Setzt Erstellungsdatum und -uhrzeit.
<i>SetLastAccessTime</i>	void	Setzt Datum und Uhrzeit des letzten Zugriffs.
<i>SetLastWriteTime</i>	void	Setzt Datum und Uhrzeit des letzten Schreibzugriffs.
<i>WriteAllBytes</i>	void	Erstellt eine neue Datei, schreibt das angegebene Byte-Array in die Datei.
<i>WriteAllLines</i>	void	Erstellt eine neue Datei, schreibt das angegebene Zeichenfolgen-Array in die Datei.
<i>WriteAllText</i>	void	Erstellt eine neue Datei, schreibt das angegebene Zeichenfolgen-Array in die Datei.

## C# Dateibearbeitung - Klasse FileInfo

---

FileInfo enthält eine Reihe von Instanzeigenschaften, die den Zustand der Datei beschreiben. So kann man beispielsweise die Länge der Datei abfragen oder sich ein Objekt vom Typ Directory zurückgeben lassen (ein Directory-Objekt beschreibt ein Verzeichnis als Objekt, ähnlich wie FileInfo eine Datei).

Eigenschaft	Beschreibung
<i>Attributes</i>	Ermöglicht das Setzen oder Auswerten der Dateiattribute ( <i>Hidden</i> , <i>Archive</i> , <i>ReadOnly</i> usw.).
<i>CreationTime</i>	Liefert oder setzt das Erstellungsdatum der Datei.
<i>Directory</i>	Liefert eine Instanz des Verzeichnisses.
<i>DirectoryName</i>	Liefert eine Zeichenfolge mit der vollständigen Pfadangabe, jedoch ohne den Dateinamen.
<i>Extension</i>	Liefert die Dateierweiterung einschließlich des vorangestellten Punktes.
<i>FullName</i>	Gibt einen String mit der vollständigen Pfadangabe einschließlich des Dateinamens zurück.
<i>LastAccessTime</i>	Liefert oder setzt die Zeit des letzten Zugriffs auf die Datei.
<i>LastWriteTime</i>	Liefert oder setzt die Zeit des letzten schreibenden Zugriffs auf die Datei.
<i>Length</i>	Gibt die Länge der Datei zurück.
<i>Name</i>	Gibt den vollständigen Namen der Datei zurück.

## C# Dateibearbeitung - Klasse FileInfo

---

Die Methoden eines Objekts vom Typ FileInfo:

Methode	Rückgabetyyp	Beschreibung
AppendText	StreamWriter	Hängt Text an eine existierende Datei an.
CopyTo	FileInfo	Kopiert die Datei an eine andere Speicherlokalität.
Create	FileStream	Erzeugt eine Datei.
CreateText	StreamWriter	Erzeugt eine neue Textdatei.
Delete	void	Löscht die Datei.
Exists	Boolean	Gibt einen booleschen Wert zurück, der false ist, wenn die angegebene Datei nicht existiert.
MoveTo	void	Verschiebt die Datei in einen anderen Ordner oder benennt sie um.
Open FileStream	void	Öffnet eine Datei.
OpenRead	FileStream	Öffnet eine Datei zum Lesen.
OpenText	StreamReader	Öffnet eine Textdatei zum Lesen.
OpenWrite	FileStream	Öffnet eine Datei zum Schreiben.

## C# Dateibearbeitung - StreamWriter

---

Dazu muss eine Datei zunächst mit einem StreamWriter verbunden werden:

```
StreamWriter myStreamWriter = new StreamWriter(@"C:\MyText.txt");
```

dem Konstruktor kann ein weiterer Parameter *true* übergeben werden, dann wird die Datei zum anhängenden Schreiben geöffnet. (bei *false* wird sie überschrieben).

**Schreiben** in den Datenstrom

```
StreamWriter sw = new StreamWriter(@"C:\NewFile.txt");  
sw.WriteLine("Visual C#");  
sw.WriteLine("macht Spaß!");  
sw.Close();
```

Die Methode `WriteLine` (wie auch die Methode `Write`) ist für jeden bekannten Datentyp vielfach überladen.

Weitere Methoden:

<i>Close</i>	Schließt das aktuelle Objekt sowie alle eingebetteten Streams.
<i>Flush</i>	Schreibt die gepufferten Daten in den Stream und löscht danach den Inhalt des Puffers.
<i>Write</i>	Schreibt in den Stream, ohne einen Zeilenumbruch anzuhängen.
<i>WriteLine</i>	Schreibt in den Stream und schließt mit einem Zeilenumbruch ab.

## C# Dateibearbeitung - StreamReader

---

Dazu muss eine Datei zunächst mit einem StreamReader verbunden werden:

```
StreamReader sr = new StreamReader(@"C:\MyTest.kkl");
```

**Lesen** aus der Datei

```
while(sr.Peek() != -1)  
    Console.WriteLine(sr.ReadLine());  
sr.Close();
```

Wichtige Methoden:

- Peek** Liest ein Zeichen aus dem Strom und liefert den das Zeichen repräsentierenden int-Wert zurück, ohne es zu verarbeiten. Der Zeiger wird nicht auf die Position des folgenden Zeichens gesetzt, wenn Peek aufgerufen wird, sondern verbleibt in seiner Stellung. *Verweist der Zeiger hinter den Datenstrom, ist der Rückgabewert -1.*
- Read** Liest ein oder mehrere Zeichen aus dem Strom und liefert den das Zeichen repräsentierenden int-Wert zurück. Ist kein Zeichen mehr verfügbar, ist der Rückgabewert -1. Der Positionszeiger verweist auf das nächste zu lesende Zeichen. Eine zweite Variante dieser überladenen Methode liefert die Anzahl der eingelesenen Zeichen.
- ReadLine** Liest eine Zeile aus dem Datenstrom – entweder bis zum Zeilenumbruch oder bis zum Ende des Stroms. Der Rückgabewert ist vom Typ string.
- ReadToEnd** Liest von der aktuellen Position des Positionszeigers bis zum Ende des Stroms alle Zeichen ein.