
Tutorial:	INI-Dateien
Letzte Änderung:	25.05.2004
Autor:	Marcel Kukelka (Chatfix)
E-Mail:	software@KuKnet.de
Web:	www.KuKnet.de
Getestet mit:	Delphi 6, Delphi 7
Copyright:	© 2002-2004 KuKnet Software
Dank an:	Pit, DeCodeGuru, MathiasSimmack (Delphi-Forum)

1. Aufbau von INI-Dateien

```
[Programm]                                1. Sektion
lastopen=20.05.2002                       Eigenschaft mit bestimmtem Wert
version=1.0                                Eigenschaft mit bestimmtem Wert

[Einstellungen]                           2. Sektion
fenster=maximiert                         Eigenschaft mit bestimmtem Wert

[User]                                    3. Sektion
username=Max Mustermann                   Eigenschaft mit bestimmtem Wert

;Kommentare beginnen mit einem Semikolon.
;Kommentare können nicht mit Delphi geschrieben oder gelesen werden.
```

INI-Dateien sind ganz normale Textdateien mit der Endung "ini".
INI-Dateien können beliebig viele Sektionen und Eigenschaften besitzen.
Die Eigenschaft bekommt ihren Wert über das = zugewiesen. Der Wert kann auch Leerzeichen enthalten.
Die Namen der Sektionen werden in [und] eingeschlossen.
Man kann INI-Dateien selber einfach im Editor anlegen bzw. bearbeiten.

2. Erste Schritte

Als erstes erweitert man die *Uses*-Liste:

```
uses {...}, IniFiles;
```

Danach benötigt man eine weitere Variable vom Typ *TIniFile*:

```
var
  {...}
  ini: TIniFile;
```

3. In INI-Datei schreiben

```
ini := TIniFile.Create('c:\MeineIni.ini');  
  // INI-Datei erstellen (falls nicht vorhanden) und initialisieren  
try  
  ini.WriteString('Sektion1', 'Eigenschaft1', 'Dein String');  
    // String in Sektion1 unter Eigenschaft1 abspeichern  
  ini.WriteInteger('Sektion2', 'Eigenschaft1', 1234);  
    // Integerwert in Sektion2 unter Eigenschaft1 abspeichern  
  ini.WriteBool('Sektion2', 'Eigenschaft2', True);  
    // Wert vom Typ Boolean abspeichern  
finally  
  ini.Free;  
    // Variable ini wieder freigeben  
end;
```

Die Befehle *WriteString*, *WriteInteger* und / oder *WriteBool* sooft hinschreiben bis man alle Daten gespeichert hat die man möchte.

Weitere Befehle zum Schreiben in eine INI-Datei:

- *WriteBinaryStream* Schreibt Daten vom Typ *TStream*
- *WriteDate* Schreibt Daten vom Typ *TDateTime*
- *WriteDateTime* Schreibt Daten vom Typ *TDateTime*
- *WriteFloat* Schreibt Daten vom Typ *Double*
- *WriteTime* Schreibt Daten vom Typ *TDateTime*

Mit folgendem Befehl erreicht man unter Windows 9x das gepufferte Daten geschrieben werden: (Anweisung zwischen *try* und *finally* schreiben)

```
ini.UpdateFile;
```

Die Methode hat unter Windows NT keine Wirkung, da NT Lese- und Schreibzugriffe auf INI-Dateien nicht puffert.

4. Aus INI-Datei lesen

Mit folgenden Anweisungen kann man Daten aus einer INI-Datei auslesen. Man benötigt hier allerdings Variablen, um die ausgelesenen Werte zu speichern.

```
var
  var1: String;
  var2: Integer;
  var3: Boolean;

{...}

ini := TIniFile.Create('c:\MeineIni.ini');
  // INI-Datei und initialisieren
try
  var1 := ini.ReadString('Sektion1', 'Eigenschaft1', '');
  // Daten von Eigenschaft1 aus Sektion1 auslesen
  var2 := ini.ReadInteger('Sektion2', 'Eigenschaft1', 0);
  // Daten von Eigenschaft1 aus Sektion2 auslesen
  var3 := ini.ReadBool('Sektion2', 'Eigenschaft2', True);
  // Daten von Eigenschaft2 aus Sektion2 auslesen
  { Der letzte Wert ist jeweils ein Standardwert wenn nicht gelesen werden kann }
finally
  ini.Free
  // Variable ini wieder freigeben
end;
```

Weitere Befehle zum Lesen aus einer INI-Datei:

- *ReadBinaryStream* Liest Daten vom Typ *TStream*
- *ReadDate* Liest Daten vom Typ *TDateTime*
- *ReadDateTime* Liest Daten vom Typ *TDateTime*
- *ReadFloat* Liest Daten vom Typ *Double*
- *ReadTime* Liest Daten vom Typ *TDateTime*

Um alle Sektionen einer INI-Datei aufzulisten benötigt man ein von *TStrings* abgeleitetes Objekt, etwa eine String-Liste, oder eine Komponenteneigenschaft, wie z.B. *Items* für eine *ListBox*-Komponente: (Anweisung zwischen *try* und *finally* schreiben.)

```
ini.ReadSections(ListBox1.Items);
```

Um alle Eigenschaften einer Sektion aufzulisten benötigt man ebenfalls ein von *TStrings* abgeleitetes Objekt: (Anweisung zwischen *try* und *finally* schreiben.)

```
ini.ReadSection('Sektion2', ListBox1.Items);
```

Um alle Werte der Eigenschaften einer Sektion aufzulisten benötigt man wieder ein von *TStrings* abgeleitetes Objekt: (Anweisung zwischen *try* und *finally* schreiben.)

```
ini.ReadSectionValues('Sektion2', ListBox1.Items);
```

5. Überprüfen ob eine Sektion / Eigenschaft existiert

Mit folgenden Anweisungen kann man überprüfen ob eine Sektion bzw. eine Eigenschaft in der betreffenden INI-Datei existiert. (Anweisungen zwischen *try* und *finally* schreiben.)

```
// existiert Sektion1 ?
if ini.SectionExists('Sektion1') then
    {...}
else
    {...}

// existiert Eigenschaft1 in Sektion1 ?
if ini.ValueExists('Sektion1', 'Eigenschaft1') then
    {...}
else
    {...}
```

6. Werte aus INI-Datei löschen

Mit folgender Anweisung kann man einen einzelnen Wert aus der INI-Datei löschen: (Anweisung zwischen *try* und *finally* schreiben.)

```
ini.DeleteKey('Sektion1', 'Eigenschaft1');
```

Mit dieser Anweisung kann man eine ganze Sektion mit allen Eigenschaften und Werten löschen: (Anweisung zwischen *try* und *finally* schreiben.)

```
ini.EraseSection('Sektion1');
```

Mit dieser Anweisung kann man die gesamte INI-Datei (oder jede beliebige andere Datei) löschen:

```
DeleteFile('c:\MeineIni.ini');
```