Registry von Martin Strohal

Einleitung

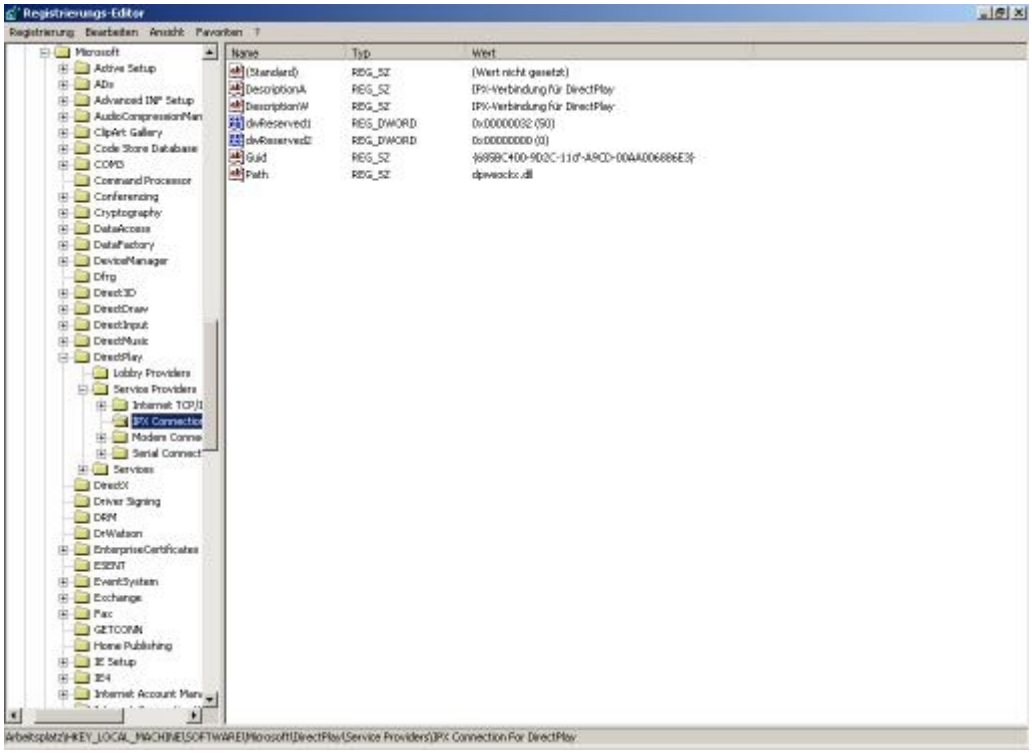
Seit Windows 95 sollten Programmeinstellungen und ähnliche "kleine" Daten nicht mehr in Ini-Dateien, sondern in der Windows-Registrierung, der Registry, gespeichert werden.

Bevor man dort allerdings Daten ablegt, sollte der Aufbau der Registry bekannt sein, da es Standards zur Erstellung von Registry-Schlüssel gibt. Damit wird dieses Tutorial beginnen.

Auf jeden Fall gilt aber: Es sollten nur Daten in der Registry geändert oder gelöscht werden, wenn man genau weiß, was man tut. Ansonsten kann es vorkommen, dass man - je nach Änderung - seine Windows-Version stark beschädigt oder gar unbrauchbar macht.

Aufbau der Registry

Theoretisches Wissen alleine ist meist nicht sehr hilfreich, folgende Erläuterungen sollten also direkt nachvollzogen werden. Die Registy besteht je nach Windows-Version aus zwei oder drei Dateien, die nicht einzeln geöffnet, sondern über einen speziellen Editor betrachtet (und bearbeitet) werden können. Dieser Editor heißt "regedit.exe" und befindet sich im Windows-Verzeichnis. Er kann also einfach gestartet werden, indem man in das Dialogfenster Start/Ausführen "regedit" eingibt.
Die Ansicht ähnelt sehr stark dem Explorer; auf der linken Seite befindet sich eine ordnerähnliche Baumstruktur, die Schlüssel (engl. Key). In der rechten Hälfte wird der Inhalt des markierten Schlüssels angezeigt. Diese Daten entsprechen den Einträgen in einer Ini-Datei; einem String-Bezeichner ist ein Wert zugeordnet.



Unterhalb des Eintrags "Arbeitsplatz" besteht die Registry aus fünf Hauptschlüsseln (Windows 2000):

- HKEY_CLASSES_ROOT (HKCR)
- HKEY_CURRENT_USER (HKCU)
- HKEY_LOCAL_MACHINE (HKLM)
- HKEY_USERS
- HKEY_CURRENT_CONFIG

In Klammern angegeben wurden die Abkürzungen, die auch von Microsoft verwendet werden.
Die Hauptschlüssel haben folgende Bedeutungen:

HKEY_CLASSES_ROOT
Hier findet die Registrierung der installierten Programme statt. Im Wesentlichen werden hier den Dateiendungen die Anwendungen zugewiesen, mit denen sie geöffnet werden können. Wie man eine solche (selbsterfundene) Dateiextension einer eigenen Anwendung zuweisen kann, wird später beschrieben.
HKEY_CLASSES_ROOT ist genau genommen ein Unterschlüssel zu HKEY_LOCAL_MACHINE\Software\CLASSES, wird in Regedit aber gesondert dargestellt.

HKEY_CURRENT_USER
Nun kommen wir zu einer der beiden Hauptrubriken, in denen Anwendungen ihre Einstellungen wie Farbauswahl, zuletzt geöffnete Dateien usw. speichern. Wie der Name sagt, betreffen die hier gespeicherten Daten nur den gerade angemeldeten Benutzer. Bei Windows NT können ja mehrere Anwender auf einem Rechner registriert sein, und jeder möchte beim Start seine eigenen Einstellungen vorfinden. Die Gesamtmenge der Einstellungen aller Nutzer werden unter HKEY_USERS gespeichert.
HKEY_CURRENT_USER enthält den Teilbereich, der für den aktuellen Nutzer zutrifft.

HKEY_LOCAL_MACHINE
Im Gegensatz zu HKCU enthält dieser Schlüssel die Daten, die für den aktuellen Rechner von Bedeutung sind - egal, wer sich angemeldet hat. Wenn eine Anwendung also z. B.

ihre Farbeinstellungen hier speichert, haben alle Anwender, die sich an diesem Rechner anmelden, die gleichen Farben. Unter HKCU dagegen könnte sich jeder Anwender seine eigenen Farben speichern.

HKEY_USERS
Hier ist die Gesamtmenge aller bekannten Benutzer gespeichert. Siehe auch HKEY_CURRENT_USER.

HKEY_CURRENT_CONFIG
Hier sind vor allem Hardware-Einstellungen wie Bildschirmauflösung, Farbtiefe und Drucker gespeichert.

In der Regel wird aus eigenen Programmen nur auf die ersten drei Schlüssel zugegriffen.

Anwendungsdaten: Wo wird gespeichert?

Die folgenden Erläuterungen treffen sowohl auf HKEY_CURRENT_USER als auch auf HKEY_LOCAL_MACHINE zu.

Wir werfen nun einen Blick auf die Unterschlüssel von HKEY_CURRENT_USER und HKY_LOCAL_MACHINE. Sie sind teilweise recht unterschiedlich, sowohl in der Anzahl als auch in der Benennung. Einen Unterschlüssel haben aber beide: Software. Und auf diesen kommt es uns an. Die anderen haben für uns bei der normalen Arbeit keine Bedeutung.

Öffnen wir nun einen der beiden "Software"-Ordner. Von den nun erscheinenden Ordnern (d.h. Unterschlüsseln) wird uns bestimmt der eine oder andere Name bekannt vorkommen. Es handelt sich in der Regel durchgehend um die Namen von Softwareherstellern. Da wir mit Delphi arbeiten, öffnen wir beispielsweise den Schlüssel "Borland". Hier finden wir nun die Namen aller installierten Borland-Produkte, z. B. auch Delphi. Unterhalb von Delphi folgt die Versionsnummer, z. B. "5.0". Das entspricht den Standardvorgaben von Microsoft.

Speichert eine Anwendung in die Registry, sollte sie folgende Schlüssel dafür verwenden:

```
HKEY_CURRENT_USER\Software\<Firma>\<Produkt>\<Version>
```

oder

```
HKEY_LOCAL_MACHINE\Software\<Firma>\<Produkt>\<Version>
```

Um die Übersichtlichkeit in der Registry zu erhalten, sollte dieses Schema beibehalten werden.

Welcher der beiden Hauptschlüssel?
Wer überlegt, in welchen der beiden Hauptschlüssel (HKCU/HKLM) er seine Daten schreiben sollte, der sollte sich für HKEY_CURRENT_USER entscheiden. Der Schlüssel ist flexibler, da jeder Benutzer seine eigenen Daten verwenden kann. Nur Daten, die unabhängig vom Anwender für alle Nutzer der Software gelten sollen, sollten unter HKLM abgelegt werden.
Auch ist zu beachten, dass unter Windows NT/2000 nur Benutzer mit Administrator-Rechten Schreibzugriff auf den HKEY_LOCAL_MACHINE-Schlüssel haben. Auch das Lesen aus diesem Schlüssel funktioniert deshalb nicht wie normal; dazu später mehr.

Schreiben in die Registry

Zuerst ein Stück Beispielcode, das die Top- und Left-Koordinaten eines Fensters in die Registry speichert:

```
uses Registry;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
var regist: TRegistry;
begin
  regist:=TRegistry.Create;
  regist.RootKey:=HKEY_CURRENT_USER;
  regist.OpenKey('Software\MeineFirma\MeinProgramm\1.0', true);
  regist.WriteInteger('Left', Form1.Left);
  regist.WriteInteger('Top', Form1.Top);
  regist.free;
end;
```

Nun der Reihe nach:
Es darf nicht vergessen werden, die Unit "Registry" in die Uses-Klausel aufzunehmen, sonst geht gar nichts.
Als erstes wird nun eine Instant mit dem Namen "regist" der Klasse TRegisty gebildet. Statt "regist" kann natürlich jeder beliebige, gültige Variablenname verwendet werden.
Als Hauptschlüssel (RootKey) wird hier HKEY_CURRENT_USER verwendet. Diese Zuweisung wäre nicht nötig, da dieser Hauptschlüssel standardmäßig verwendet wird, wenn RootKey nichts zugewiesen wird - aber sicher ist sicher.
Nun wird der Schlüssel geöffnet, in den wir unsere Daten schreiben wollen. Der Boolean-Wert (zweiter Parameter von OpenKey) hat folgende Bedeutung: Bei "true" wird der Schlüssel erzeugt, wenn er noch nicht existiert, bei "false" nicht.
Mit den Methoden WriteInteger werden die Daten gespeichert. Der erste Wert gibt den Namen an, unter dem der folgende Wert (zweiter Parameter) später wieder abrufbar ist. Bei "WriteInteger" muss der zu speichernde Wert natürlich eine Ganzzahl sein. Daneben gibt es für andere Datentypen entsprechende Methoden wie "WriteString", "WriteDateTime", "WriteFloat" usw.
Wird der Registry-Zugriff nicht mehr benötigt, sollte er geschlossen werden (regist.free). Damit wird der Speicherplatz der Instanz "regist" im Arbeitsspeicher wieder freigegeben. Wäre stattdessen ein Zugriff auf einen anderen Schlüssel nötig gewesen, hätte der aktuelle über "CloseKey" geschlossen werden können.

Lesen aus der Registry

Das Lesen von Daten aus der Registry gestaltet sich nicht sehr viel anders als das Schreiben. Statt der Write-Methoden werden die Read-Methoden verwendet. Das Ganze sieht dann so aus:

```
uses Registry;


procedure TForm1.FormCreate(Sender: TObject);
var regist: TRegistry;
begin
  regist:=TRegistry.Create;
  regist.RootKey:=HKEY_CURRENT_USER;
  regist.OpenKey('Software\MeineFirma\MeinProgramm\1.0', true);
  Form1.Left:=regist.ReadInteger('Left');
  Form1.Top:=regist.ReadInteger('Top');
  regist.free;
```

```
end;
```

Den Read-Methoden (hier ReadInteger) wird in Klammern der Name des Wertes übergeben, der ausgelesen werden soll. Dieser Wert wird dann, wie bei Funktionen üblich, als Ergebniswert zurückgeliefert.

Nun könnte es sein, dass es sich beim Ausführen obiger Methode um den ersten Programmstart handelt, der Schlüssel MeineFirma\MeinProgramm\1.0 also noch gar nicht existiert. Da wir bei OpenKey als zweiten Parameter "true" angegeben haben, wird der Schlüssel dadurch erzeugt. Die Werte "Left" und "Top" gibt es aber nach wie vor nicht und können deshalb auch nicht ausgelesen werden. In solchen Fällen löst die Laufzeitumgebung eine Exception (ERegistryException) ausgelöst. Deshalb sollten Auslese-Methoden immer innerhalb eines try-except-Blocks stehen:

```
uses Registry;

procedure TForm1.FormCreate(Sender: TObject);
var regist: TRegistry;
begin
  regist:=TRegistry.Create;
  regist.RootKey:=HKEY_CURRENT_USER;
  try
    regist.OpenKey('Software\MeineFirma\MeinProgramm\1.0', true);
    Form1.Left:=regist.ReadInteger('Left');
    Form1.Top:=regist.ReadInteger('Top');
  except
    Form1.Left:=0;
    Form1.Top:=0;
  end;
  regist.free;
end;
```

Der Except-Abschnitt wird nur durchlaufen, wenn die Werte nicht ausgelesen werden können. Stattdessen werden dann die Standardwerte Left:=0 und Top:=0 verwendet.

Besonderheiten bei Windows NT/2000
Wie bereits erwähnt, können unter NT-Systemen nur Anwender mit Administrator-Rechten auf den HKEY_LOCAL_MACHINE-Schlüssel der Registry *schreibend* zugreifen. Wird also in einem Programm der Schreibzugriff auf diesen Schlüssel verwendet, sollte try-except verwendet werden, um ein mögliches Fehlschlagen abzufangen.
Auch das Lesen aus diesem Schlüssel ist unter NT nicht ganz problemlos. Über die oben vorgestellte Möglichkeit versucht Delphi immer, Vollzugriff (lesen *und* schreiben) zu erlangen. Ist das nicht möglich, weil ein NT-User keine Administrator-Rechte hat, tritt eine Exception auf. Da das Lesen von HKLM jedoch allen NT-Usern möglich ist, müssen wir Delphi klar machen, dass wir in diesem Fall keinen Vollzugriff, sondern nur Lesezugriff wollen. Das geht so:

```
  regist:=TRegistry.Create(KEY_READ);
```

Weitere Möglichkeiten zum Thema Zugriffsrechte sind in der Delphi-Hilfe unter TRegistry.Access zu finden.

Spezielle Schlüssel in der Registry

Wer etwas in der Registry stöbert, findet einige interessante Schlüssel, die auch von eigenen Anwendungen ausgelesen werden können.
Gutes Beispiel ist der Schlüssel

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion. Zu beachten ist nur, dass unter NT statt "Windows" der Schlüssel "Windows NT" verwendet werden muss.

CurrentVersion	Windows-Version, z.B. 5.0
ProductName	Klartext Windows-Version, z.B. Microsoft Windows 2000
RegisteredOwner	Name des Besitzers
RegisteredOrganization	Name der Firma
SourcePath	Verzeichnis, aus dem Windows installiert wurde (meist CD)
PathName	Windows-Verzeichnis

Autostart
Auch interessant ist der Schlüssel HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run (heißt auch bei NT so). Hier können Anwendungen eingetragen werden, die bei jedem Windows-Start ausgeführt werden sollen. Es muss sich um einen Stringwert handeln, der auf eine ausführbare Datei (inkl. Pfad) verweist. Der Bezeichner dafür ist beliebig.
Neben dem Schlüssel "Run" gibt es auch "RunOnce". Programme, die hier aufgeführt werden, werden nur beim nächsten Windows-Start ausgeführt und dann aus dem Autostart gelöscht.

Eigene Dateiendungen registrieren

Jeder kennt das: Legt man in einer Anwendung eigene Dateien an, bekommen diese eine bestimmte Endung, über die ihr Typ zu erkennen ist. DOC ist z.B. ein Word-Dokument. Über Doppelklick im Explorer wird dann automatisch die richtige Anwendung gestartet und die Datei darin geöffnet. Dazu ist Folgendes nötig:

Als erstes denken wir uns eine Dateiendung aus (muss nicht mehr unbedingt dreistellig sein, auch vierstellig ist möglich) - diese Endung sollte auf jeden Fall nicht bereits weit verbreitet, sondern auf den Zielrechnern möglichst eindeutig sein.
Zur Registrierung der Endung (nehmen wir als Beispiel "ABC") muss Folgendes in die Registry eingetragen werden:

```
HKEY_CLASSES_ROOT\.abc
```

Vor der Endung abc muss ein Punkt stehen. Ist dieser Schlüssel angelegt, muss sein Standardwert verändert werden. Schaut man sich die Einträge in der Registry an, sieht man, dass jeder Schlüssel als ersten Eintrag einen Wert namens "(Standard)" besitzt, meist mit dem Inhalt "(Wert nicht zugeordnet)". Um aus Delphi heraus diesen Standardwert zu verändern, darf als Bezeichner nicht "(Standard)" angegeben werden, sondern der Bezeichner muss ein leerer String sein:

```
regist.WriteString('', 'MeinProgramm');
```

Das liegt daran, dass „(Standard)" nur ein Anzeigewert ist. Der eigentliche Schlüsselname ist ein Leerstring, den man bei der Anzeige aber schlecht sieht.
Bei dem Wert, der dem Standardeintrag zugeordnet wird, muss es sich ebenfalls um einen eindeutigen String handeln. Scrollt man in der Registry im Schlüssel HKEY_CLASSES_ROOT weiter nach unten, so folgen nach den Endungen, die alle mit einem Punkt beginnen, diese Bezeichner. Wie folgt sind nun Schlüssel und Werte anzulegen:

```
HKEY_CLASSES_ROOT\MeinProgramm
          (Standard)="Dateityp-Beschreibung"
HKEY_CLASSES_ROOT\MeinProgramm\DefaultIcon
          (Standard)="c:\test.exe,1"
HKEY_CLASSES_ROOT\MeinProgramm\Shell\Open
          (Standard)="Öffnen"
HKEY_CLASSES_ROOT\MeinProgramm\Shell\Open\Command
          (Standard)="c:\test.exe %1"
```

Was hier mit "Dateityp-Beschreibung" angegeben ist, ist der Text, der auch im Explorer angezeigt wird, wenn Detailansicht eingestellt ist.
DefaultIcon ist einfach der vollständige Pfad auf ein Symbol, das allen Dateien mit der Endung abc zugeordnet werden soll. Falls sich das Symbol nicht in einer getrennten ico-Datei befindet, sondern in der EXE-Datei, kann über Nummerierung darauf zugegriffen werden wie oben.
Die dritte Zeile definiert das Kommando "Öffnen". Der beschreibende Text stellt den obersten Eintrag des Kontextmenüs im Explorer dar, wenn mit der rechten Maustaste auf den Dateinamen geklickt wird.
Als letztes schließlich der Kommandozeilenbefehl, um die eigene Anwendung zu öffnen. %1 bewirkt, dass der Name der Datei, die mit der Anwendung geöffnet werden soll, der Anwendung als Parameter übergeben wird. Wird %1 weggelassen, wird bei jedem Doppelklick auf eine abc-Datei zwar test.exe gestartet, die abc-Datei jedoch nicht geöffnet.
%1 alleine reicht natürlich auch nicht. Damit hat man den Dateinamen jedoch im Programm und kann dort im OnCreate-Ereignis über ParamCount und ParamStr prüfen, ob ein Dateiname übergeben wurde und diese Datei dann laden.
Mehr zur Behandlung von Parametern in unserer Rubrik "Tipps & Tricks", Abschnitt "Object Pascal".

Neben einem "Open"-Kommando kann auch ein "Print"-Kommando festgelegt werden, so dass Dateien mit dieser Endung direkt aus dem Kontextmenü im Explorer ausgedruckt werden können.

Ausblick

Das war das Wichtigste, was beim Arbeiten mit der Registry bekannt sein sollte.
Natürlich bietet Delphi für das Objekt TRegistry weitere Methoden an, um z. B. zu prüfen, ob ein Schlüssel weitere Unterschlüssel hat (HasSubKeys) oder ob ein bestimmter Wert vorhanden ist (ValueExists) und auch zum Verschieben von ganzen Schlüsseln (MoveKey). Näheres zu diesen vielen Möglichkeiten ist in der Delphi-Hilfe unter TRegistry zu finden. Wenn das hier dargestellte Grundwissen bekannt ist, dürfte eine Verwendung dieser weiteren Methoden nicht sehr schwer fallen.