Denne guide er oprindeligt
udgivet på Eksperten.dk

COMPUTERWORLD

Register-databasen med .NET

Mange programmer gemmer deres konfiguration-data i register-database. Denne artikel viser hvordan du - f.eks. vha. C# - henter værdier fra registeret, hvordan du indsætter nye værdier, og hvordan du sletter eksisterende.

Skrevet den **02. Feb 2009** af **nielle** I kategorien **Programmering / C#** | ★★★☆☆

Indledning

Programmer har brug for at kunne gemme data om deres konfiguration.

Gennem Windows' historie har der været forskellige måder at gøre det på. Først var det som INI-filer, men med Windows 9x gik man mere og mere over til at bruge register-databasen, og senest med introduktionen af .NET anbefales det at man bruger XML-filer (config-filer) til formålet.

Selv om brug af register-databasen altså er "gammel teknologi" i forhold til .NET, er den alligevel ganske vigtig. Dels er der masse eksisterende programmer som stadig bruger den, og dels er det ikke alle som har taget den nye XML-teknologi til sig.

Denne artikel handler om hvordan man browser rundt i register-databasen, hvordan man opretter register-nøgler og værdifelter, og hvordan man sletter dem igen.

Det er i øvrigt min plan at jeg senere kommer men en artikel om hvordan man egentlig håndtere konfiguration "the .NET way".

v. 1.0: 28/01/2008 - Første version.

Kort om notationen

I forbindelse med register-databasen snakker man om register-nøgler (engelsk: *registry keys*) og værdier (*values*).

I min mening er især ordet "values" lidt uheldigt valgt; et bedre ord ville have været noget i stil med "register-variable". En værdi har nemlig et navn og så altså - ja - en værdi. For at kunne skelne mellem selve værdien, og dens værdi, har jeg valgt at kalde dem for *værdifelter* i det følgende. Et værdifelt har altså et navn og en værdi.

Register-nøgler og værdifelter er, i analogi med biblioteker og filer i filsystemet, organiseret i en træstruktur (analogi: "et drev"). Sådan et træ kaldes i register-databasen for et *hive* (dansk: *bistade* - hmmm, gad vide hvorfor man kalder den det?)..

Hive's

Nu er dette ikke direkte en artikel om register-databasen som sådan, men den skal da forklares kort.

Hvis man ønsker at kigge lidt rundt og se hvad der er i register-databasen, kan man f.eks. bruge regedit.exe. Det kommer med Windows og kan startes direkte fra Kør-menupunktet i Windows' Start-

menu.

Set med regedit består register-databasen umiddelbart af 5 træer (hive's):

```
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS
HKEY_CURRENT_CONFIG
```

(Der er muligvis flere i din Windows installation)

Tre af dem er imidlertid genveje til grene længere oppe i træerne:

HKEY_CLASSES_ROOT er en genvej til HKEY_LOCAL_MACHINE\Software\Classes
HKEY_CURRENT_USER er en genvej til en mappe under HKEY_USERS.
HKEY_CURRENT_CONFIG er en genvej til

Træerne er bygget op af mapper - register-nøgler. I mapperne kan der være undermapper og der kan være værdifelter (og i øvrigt begge dele på samme tid).

Når man har brug for at bevæge sig rundt i register-databasen, starter man med at vælge det rigtige hive og så bevæge sig ind i mapperne indtil at man har fundet den værdi man ønsker.

Dette gør man vha. Register-klassen, som har et field for hvert muligt hive:

```
Registry.ClassesRoot     - HKEY_CLASSES_ROOT
Registry.CurrentConfig   - HKEY_CURRENT_CONFIG
Registry.CurrentUser     - HKEY_CURRENT_USER
Registry.DynData         - HKEY_DYN_DATA (kun i Windows 98/Windows Me)
Registry.LocalMachine    - HKEY_LOCAL_MACHINE
Registry.PerformanceData - HKEY_PERFORMANCE_DATA
Registry.Users           - HKEY_USERS
```

Browse i register-databasen

Lad os straks hoppe i med et eksempel (som dog kun giver mening for folk som har Visual Studio installeret):

```
using System;
using Microsoft.Win32;  // Der er her man finder registry-klasserne

namespace RegistryDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            // Start i HKEY_CURRENT_USER.
            RegistryKey regKey = Registry.CurrentUser;
```

```
            Console.WriteLine("Register-nøgle : " + regKey.Name);

            // Åbn mapperne Software -> Microsoft -> VisualStudio,
            regKey = regKey.OpenSubKey(@"Software\Microsoft\VisualStudio");
            Console.WriteLine("Register-nøgle : " + regKey.Name);

            // Hent navnene på underfolderne af VisualStudio.
            string[] subRegKeyNameArr = regKey.GetSubKeyNames();

            // Løb underfolderne igennem.
            foreach (string subRegKeyName in subRegKeyNameArr)
            {
                // Åbn underfolderen.
                RegistryKey subRegKey = regKey.OpenSubKey(subRegKeyName);
                Console.WriteLine("Register-nøgle : " + subRegKey.Name);

                Console.WriteLine("\tRegister-nøglens navn : " +
subRegKeyName);

                // Hvilken version af Visual Studio?
                Console.Write("\t");
                switch (subRegKeyName)
                {
                    case "6.0":
                        Console.WriteLine("Microsoft Visual Studio 6.0");
                        break;
                    case "7.0":
                        Console.WriteLine("Microsoft Visual Studio .NET
2002");
                        break;
                    case "7.1":
                        Console.WriteLine("Microsoft Visual Studio .NET
2003");
                        break;
                    case "8.0":
                        Console.WriteLine("Microsoft Visual Studio 2005");
                        break;
                    case "9.0":
                        Console.WriteLine("Microsoft Visual Studio 2008");
                        break;
                    case "Debugger":
                        Console.WriteLine("Debugger");
                        break;
                    default:
                        Console.WriteLine("Ukendt version af VS");
                        break;
                }

                // Antal undernøgler og værdier
                int subSubKeyCount = subRegKey.SubKeyCount;
                int subSubValueCount = subRegKey.ValueCount;
                Console.WriteLine("\tUndernøgler: {0}; Værdi-felter {1}",
                    subSubKeyCount, subSubValueCount);

                // Hent stien for den seneste solution.
```

```
                object lastLoadedSolutionObj =
subRegKey.GetValue("LastLoadedSolution");
                Console.Write("\t");
                if (lastLoadedSolutionObj != null)
                    Console.WriteLine("Seneste solution : " +
lastLoadedSolutionObj.ToString());
                else
                    Console.WriteLine("Seneste solution : <udefineret>");
            }
        }
    }
}
```

Et typisk output kunne være:

```
Register-nøgle : HKEY_CURRENT_USER
Register-nøgle : HKEY_CURRENT_USER\Software\Microsoft\VisualStudio
Register-nøgle : HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\7.1
    Register-nøglens navn : 7.1
    Microsoft Visual Studio .NET 2003
    Undernøgler: 21; Værdi-felter 26
    Seneste solution : C:\Source.Net11\FormatTxt\FormatTxt.sln
Register-nøgle : HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\8.0
    Register-nøglens navn : 8.0
    Microsoft Visual Studio 2005
    Undernøgler: 61; Værdi-felter 24
    Seneste solution : C:\Source.Net11\FormatTxt\e816589.sln
Register-nøgle : HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\9.0
    Register-nøglens navn : 9.0
    Microsoft Visual Studio 2008
    Undernøgler: 48; Værdi-felter 22
    Seneste solution : C:\Source.Net35\RegistryDemo\RegistryDemo.sln
Register-nøgle : HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\Debugger
    Register-nøglens navn : Debugger
    Debugger
    Undernøgler: 1; Værdi-felter 0
    Seneste solution : <udefineret>
```

Kommentarerne forklare forhåbentligt det meste, men her er lidt mere forklaring om hvad der sker i koden:

Register-nøgler håndteres generelt med RegistryKey klassen.

Med Registry.CurrentUser angives det at koden skal starte i HKEY_CURRENT_USER hive't. Registry klassen har som sagt tilsvarende værdier for de andre hive's.

Med OpenSubKey() metoden navigeres der hen til den ønskede register-nøgle; her er det folderen som indeholder oplysningerne om hvordan Visual Studio er installeret og konfigureret.

Enhver register-nøgle kan indeholde mange undernøgler. Man får en liste over disses navne med

GetSubKeyNames(). I koden benyttes dette til at "skanne" igennem VS mappen og de enkelte VS installationer. Der findes en tilsvarende funktion til at få værdifelterne - GetValuesNames().

Med hhv. SubKeyCount og ValueCount property'erne får man at vide hvor mange undernøgler (altså mapper) og hvor mange værdifelter der er i mappen.

Endeligt får man fat på en bestemt værdi med GetValue() metoden.

OpenRemoteBaseKey()

OpenSubKey() åbner til en register-nøgle på den lokale maskine. Man kan faktisk også åbne til en register-nøgle på en remote maskine (hvis man har de rette privilegier).

Dette gøres med:

```
RegistryKey.OpenRemoteBaseKey(RegistryHive.CurrentUser, "machineName")
```

RegistryHive er en enum, som har de samme værdier som Registry. Man kan undre sig lidt over at man ikke har anvendt den samme enum til både lokal og remote brug.

Ud over at det foregår på en remote computer, er fremgangen ellers stort set den samme. Jeg vil henvise til online dokumentationen for de finere detaljer omkring f.eks. privilegier.

Oprette register-nøgler

Ovenstående kode browser kun igennem de register-nøgler og værdifelter der allerede eksisterer. Man hvad nu hvis vi selv ønsker at oprette nye nøgler og værdier? Eller alternativt at ændre på nogen af de eksisterende værdier?

NB: Inden at du begynder at oprette, modificere, eller slette register-nøgler og værdifelter, skal du være opmærksom på at du skal passe på. Register-databasen indeholder konfiguration oplysninger for mange af programmerne på din maskine inkl. Windows selv. Så pas på at du ikke kommer til - uforvarende - at slette eller overskrive disse oplysninger!

Først og fremmest. Hvis man ønsker at oprette register-nøgler, som hører til et program man selv har skrevet, er det kostume at disse ligges under:

HKEY_CURRENT_USER\Software

eller:

HKEY_LOCAL_MACHINE\Software

Hvis det et program som er specifikt installeret for brugeren hhv. til alle brugerne på maskinen.

Normalt vil OpenSubKey() åbne en registraturnøgle i readonly-tilstand. Hvis man ønsker at oprette registratur nøgler og værdier, skal den have et argument mere:

Opret en register-nøgle:

```
RegistryKey regKey = Registry.LocalMachine;

// Åbn i skrivbar form.
regKey = regKey.OpenSubKey("Software", true);

// Opret register-nøglen "Pibgorn".
regKey.CreateSubKey("Pibgorn");
```

Man kan oprette undernøgler i én arbejdsgang med:

```
// Opret nøgle og undernøgler.
regKey.CreateSubKey(@"Pibgorn\Geoff\Drusilla");
```

eller:

```
// Opret nøgle og undernøgler.
regKey = regKey.CreateSubKey("Pibgorn");
regKey = regKey.CreateSubKey("Geoff");
regKey.CreateSubKey("Drusilla");
```

Oprette værdifelter

Værdifelter oprettes, eller opdateres, med SetValue():

Opret et værdifelt:

```
// Opret nøglen.
regKey = regKey.CreateSubKey("Pibgorn");

// Opret et værdifelt med navnet "Thorax" og værdien "7, 9, 13".
regKey.SetValue("Thorax", "7, 9, 13");
```

Værdityper

Register-databasen har forskellige værdityper på samme måde som C# har variabel-typer.

Argument nr. 2 i SetValue() tager et objekt, og der vælges så automatisk den register-type, der passer bedst. Hvis objektet er af en type som ikke er understøttet - f.eks. en int[] - smides der en ArgumentException i stedet.

I det ovenstående tilfælde vælger Register-basen selv at gemme som en REG_SZ da dette er den type som

passer bedst. Man kan også angive typen direkte som 3. argument, og på den måde bestemme hvilken type man ønsker at gemme værdien som.

De mulige typer er:

```
Type          - Regedit.exe menu navn - RegistryValueKind enum - C# type

REG_BINARY    - Binær værdi           - .Binary                - byte[]
REG_DWORD     - DWORD-værdi           - .DWord                 - int (Int32)
REG_EXPAND_SZ - Udvidelig strengværdi - .ExpandString          - string
REG_MULTI_SZ  - Multistrengsværdi     - .MultiString           - string[]
REG_QWORD     - <kan ikke vælges>     - .QWord                 - long (Int64)
REG_SZ        - Strengværdi           - .String                - string
...           - ...                   - .Unknown               - ...
```

Typen af et værdifelt aflæses med GetValueKind().

Typerne i aktion:

```
RegistryKey regKey = Registry.LocalMachine;

// Åbn i skrivbar form.
regKey = regKey.OpenSubKey("Software", true);

// Opret nøglen "Nielle".
regKey = regKey.CreateSubKey("Nielle");

// Opret et antal værdifelter under "Nielle".

// Som REG_BINARY.
byte[] binary = new byte[] { 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 };
regKey.SetValue("Binary", binary, RegistryValueKind.Binary);

// Som REG_DWORD.
int dWord = 42;
regKey.SetValue("DWord", dWord, RegistryValueKind.DWord);

// Som REG_EXPAND_SZ.
string expandString = "abc";
regKey.SetValue("ExpandString", expandString, RegistryValueKind.ExpandString);

// Som REG_MULTI_SZ.
string[] multiString = new string[] { "ælle", "bælle", "mig", "fortælle" };
regKey.SetValue("MultiString", multiString, RegistryValueKind.MultiString);

// Som REG_QWORD.
long qWord = 1123581321;
regKey.SetValue("QWord", qWord, RegistryValueKind.QWord);

// Som REG_SZ.
string str = "Leonardo da Vinci";
regKey.SetValue("String", str, RegistryValueKind.String);
```

```
// Gennemløb de oprettede værdifelter.
string[] valueNameArr = regKey.GetValueNames();
foreach (string valueName in valueNameArr)
{
    // Værdifeltets register-type.
    RegistryValueKind valueKind = regKey.GetValueKind(valueName);

    // Værdifeltets værdi.
    object value = regKey.GetValue(valueName);

    Console.WriteLine("{0} - {1} - {2}", valueName, valueKind,
value.GetType().ToString());
}
```

REG_EXPAND_SZ

REG_EXPAND_SZ og REG_SZ er begge strenge, men REG_EXPAND_SZ har yderlig den egenskab at indlejrede environment variable ekspanderes når værdien hentes fra Registratуret:

```
string expandString = "ABC %TEMP% DEF";
regKey.SetValue("ExpandString", expandString, RegistryValueKind.ExpandString);

string echo = regKey.GetValue("ExpandString") as string;
Console.WriteLine(echo);

string defaultValue = "Findes ikke";
echo = regKey.GetValue("ExpandString", defaultValue,
    RegistryValueOptions.DoNotExpandEnvironmentNames) as string;
Console.WriteLine(echo);
```

(Standard) værdien

Enhver register-nøgle har pr. default et værdifelt fra starten af. I regedit.exe optræder det som "(Standard)" og dets værdi er tom. Dens navn er i øvrigt også tomt, og derfor så betjenes den via den tomme streng:

Sæt standard-værdien:

```
// Opret nøglen "Pibgorn".
regKey = regKey.CreateSubKey("Pibgorn");

// Sæt standard-værdien til "Gaggot".
regKey.SetValue("", "Gaggot");

string std = regKey.GetValue("") as string;
Console.WriteLine(std);
```

Der kan naturligvis kun være et standard værdifelt pr. register-nøgle.

Opdatering af værdifelt

Hvis man bruger SetValue() på et værdifelt, som allerede eksistere, bliver værdien af dette simpelthen opdateret med den nye værdi.

Sletning af værdifelter

Et værdifelt slettes fra den register-nøgle hvor den sidder:

Sletning af et værdifelt:

```
// Værdifeltet oprettes/opdateres.
regKey.SetValue("valueName", 123);

// ...

// Værdifeltet slettes.
regKey.DeleteValue("valueName");
```

Hvis der ikke er noget værdifelt med det pågældende navn på register-nøglen, resultere dette i en ArgumentException. Denne kan man enten vælge at try-catch'e, eller man kan vælge helt at undertrykke den med:

```
// Værdifeltet slettes - uanset om det overhovedet eksistere.
regKey.DeleteValue("valueName", false);
```

Sletning af register-nøgler

Register-nøgler slettes med DeleteSubKey() eller DeleteSubKeyTree(). DeleteSubKey() sletter enkelt-nøgler uanset om de indeholder værdifelter eller ej:

Sletning af en register-nøgle:

```
// Opret nøglen "Pibgorn" med en enkelt undernøgle.
RegistryKey subRegKey = regKey.CreateSubKey("Pibgorn");
RegistryKey subSubRegKey = subRegKey.CreateSubKey("Prince Crewth");

// ...

// Opret et værdifelt.
subSubRegKey.SetValue("Luciano", 7 - 9 - 13);

// ...
```

```
// Register-nøglen slettes.
subRegKey.DeleteSubKey("Prince Crewth");
```

Ligesom ved DeleteValue() vil der blive smidt en exception hvis man forsøger at slette en register-nøgle som ikke findes. Og lige som ved DeleteValue() kan denne undertrykkes.

```
subRegKey.DeleteSubKey("eksistereIkke", false);
```

Der smides under alle omstændigheder en InvalidOperationException, hvis man forsøger at slette en register-nøgle som indeholder undernøgler. For at slette disse skal man bruge DeleteSubKeyTree() i stedet.

Sletning af en register-nøgle med undernøgler:

```
// Opret nøglen "Pibgorn", og et "træ" af undernøgler.
regKey = regKey.CreateSubKey("Pibgorn");
RegistryKey subRegKey = regKey.CreateSubKey("Prince Crewth");
RegistryKey subSubRegKey = subRegKey.CreateSubKey("Gaggot");

// ...

// Slet register-nøglen, alle dens
// undernøgler og alle dens værdifelter.
regKey.DeleteSubKeyTree("Prince Crewth");
```

En utility klasse

Jeg vil afslutte artiklen med denne lille utility-klasse som er yderst praktisk hvis man blot ønsker at kunne sætte og aflæse værdifelter:

```
public class RegistryConfig
{
    private RegistryKey rootRegKey;

    public RegistryConfig(string rootName)
    {
        rootRegKey = Registry.LocalMachine;
        rootRegKey = rootRegKey.OpenSubKey("Software", true);

        // Opretter (non-destruktivt) en register-nøgle.
        rootRegKey = rootRegKey.CreateSubKey(rootName);
    }

    // Henter og sætter værdien af et værdifelt.
    public object this[string name]
```

```
    {
        get { return rootRegKey.GetValue(name); }
        set { rootRegKey.SetValue(name, value); }
    }

    public void SetDefaultValue(string name, object defaultValue)
    {
        SetDefaultValue(name, defaultValue, false);
    }

    // Initialisere et værdifelt, hvis det ikke allerede er sat.
    public void SetDefaultValue(string name, object defaultValue, bool reset)
    {
        if (this[name] == null || reset)
        {
            this[name] = defaultValue;
        }
    }
}
```

Et eksempel på hvordan den kan bruges:

```
RegistryConfig cfg = new RegistryConfig("Pibgorn");

// Sæt et par værdier
cfg["Pibgorn"] = "Geoff";  // Bliver gemt som REG_SZ
cfg["Drusilla"] = 7 - 9 - 13;  // Bliver gemt som REG_DWORD

Console.WriteLine(cfg["Pibgorn"]);
Console.WriteLine(cfg["Drusilla"]);

cfg.SetDefaultValue("slut", "fin");
Console.WriteLine(cfg["slut"]);
```

Andre links

http://www.csharphelp.com/archives2/archive430.html

Feedback

cool_code, det er fint at du ikke synes super om artiklen. Men du må da meget gerne komme med lidt konstruktiv kritik så. :^)

Kommentar af cool_code d. 06. Feb 2008 | 1

Kommentar af mysitesolution d. 05. Mar 2008 | 2

God artikel. Desværre er registreringsdatabasen, som du siger, vigtig :(Til fx filformater. Lidt trist

Kommentar af cwboy d. 01. Feb 2008 | 3

God gennemgang, med mange gode eksempler.