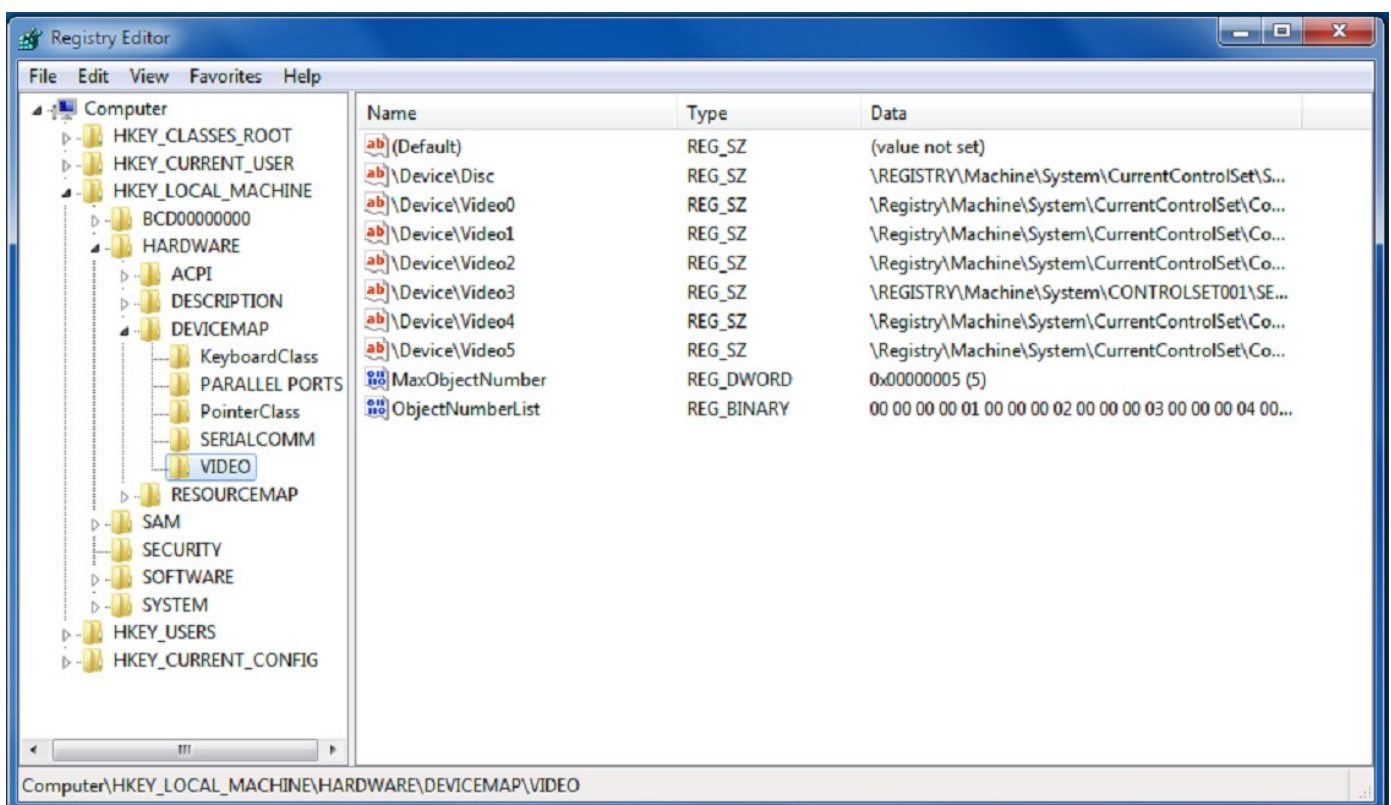


رجیستری یک پایگاه داده‌ی سیستمی است که برنامه‌ها، اجزای سیستم و اطلاعات پیکربندی در آن ذخیره و بازیابی می‌شود. داده‌های ذخیره شده در رجیستری مطابق با نسخه ویندوز فرق می‌کنند. نرم‌افزارها برای بازیابی، تغییر و پاک کردن رجیستری از API های مختلفی استفاده می‌کنند. خوشبختانه .NET نیز امکانات لازم برای مدیریت رجیستری را فراهم کرده است.

در صورت رخداد خطا در رجیستری، امکان خراب شدن ویندوز وجود دارد در نتیجه با احتیاط عمل کنید و قبل از هر کاری از رجیستری پشتیبان تهیه نمایید. قبل از شروع به کدنویسی قدری با ساختار رجیستری آشنا شویم تا در ادامه قادر به درک مفاهیم باشیم.

## ساختار رجیستری

رجیستری اطلاعات را در ساختار درختی نگاه می‌دارد. هر گره در درخت، یک کلید (key) نامیده می‌شود. هر کلید می‌تواند شامل چندین زیرکلید (subkey) و چندین مقدار (value) باشد. در برخی موارد، وجود یک کلید تمام اطلاعاتی است که نرم افزار بدان نیاز دارد و در برخی موارد، برنامه کلید را باز کرده و مقادیر مربوط به آن کلید را می‌خواند. یک کلید می‌تواند هر تعداد مقدار داشته باشد و مقادیر به هر شکلی می‌توانند باشند. هر کلید شامل یک یا چند کاراکتر است. نام کلیدها نمی‌توانند کاراکتر “\” را داشته باشند. نام هر زیرکلید یکتاست و وابسته به کلیدی است که در سلسله مراتب، بلافاصله بالای آن می‌آید. نام کلیدها باید انگلیسی باشند اما مقادیر را به هر زبانی می‌توان نوشت. در زیر یک نمونه از ساختار رجیستری را مشاهده می‌کنید که در نرم‌افزار registry editor به نمایش در آمده است.



هر کدام از درخت‌های زیر my computer یک کلید است. HKEY\_LOCAL\_MACHINE دارای زیرکلیدهایی مثل SAM ، HARDWARE و SECURITY است. هر مقدار شامل یک اسم، نوع و داده‌های درون آن است. برای مثال MaxObjectNumber از مقادیر زیرکلید HKEY\_LOCAL\_MACHINE\HARDWARE\DEVICEMAP\VIDEO است. داده‌های درون هر مقدار می‌تواند از انواع باینری، رشته‌ای و عددی باشد؛ برای مثال MaxObjectNumber یک عدد ۳۲ بیتی است.

محدودیت‌های فنی برای نوع و اندازه‌ی اطلاعاتی که در رجیستری ذخیره می‌گردد، وجود دارد. برنامه‌ها باید اطلاعات اولیه و پیکربندی را در رجیستری نگه دارند و سایر داده‌ها را در جای دیگر ذخیره کنند. معمولاً داده‌های بیش‌تر از یک یا دو کیلوبایت باید در یک فایل ذخیره شوند و با استفاده از یک کلید در رجیستری به آن فایل رجوع کرد. برای حفظ فضای ذخیره سازی باید داده‌های شبیه به هم در یک ساختار جمع آوری گردند و ساختار را به عنوان یک مقدار ذخیره کرد؛ به جای آن که هر عضو ساختار را به عنوان یک کلید ذخیره کرد. ذخیره سازی اطلاعات به صورت باینری این امکان را می‌دهد که اطلاعات را در یک مقدار ذخیره کنید.

اطلاعات رجیستری در پیچ فایل ( Page File ) ذخیره می‌شوند. پیچ فایل ناحیه‌ای از حافظه RAM است که می‌تواند در زمانی که استفاده نمی‌شود به Hard منتقل شود. اندازه‌ی پیچ فایل به وسیله‌ی مقدار PagedPoolSize در کلید HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management مطابق با جدول زیر تنظیم می‌گردد.

مقدار	توضیحات
0x00000000	سیستم یک مقدار بهینه را تعیین می‌کند
0x1-0x20000000	یک اندازه مشخص برحسب بایت که در این بازه باشد
0xFFFFFFFF	سیستم بیش‌ترین مقدار ممکن را تشخیص می‌دهد

### کلیدهای از پیش تعریف شده

یک برنامه قبل از آن که اطلاعاتی را در رجیستری درج کند باید یک کلید را باز کند. برای باز کردن یک کلید می‌توان از سایر کلیدهایی که باز هستند، استفاده کرد. سیستم کلیدهایی را از پیش تعریف کرده که همیشه باز هستند. در ادامه کلیدهای از پیش تعریف شده را قدری بررسی می‌کنیم.

#### HKEY\_CLASSES\_ROOT

زیرشاخه‌های این کلید، انواع اسناد و خصوصیات مربوط به آن‌ها را مشخص می‌کنند. این شاخه نباید در یک سرویس یا برنامه‌ای که کاربران متعدد دارد، مورد استفاده قرار گیرد.

#### HKEY\_CURRENT\_USER

زیرشاخه‌های این کلید، تنظیمات مربوط به کاربر جاری را مشخص می‌کنند. این تنظیمات شامل متغیرهای محیطی، اطلاعات درباره‌ی برنامه‌ها، رنگ‌ها، پرینترها، ارتباطات شبکه و تنظیمات برنامه‌هاست. به طور مثال میکروسافت اطلاعات مربوط به برنامه‌های خود را در کلید HKEY\_CURRENT\_USER\Software\Microsoft ذخیره می‌کند. هر کدام از برنامه‌ها یک زیرکلید در کلید مزبور را به خود اختصاص داده‌اند. این شاخه نیز نباید در یک سرویس یا برنامه‌ای که کاربران متعدد دارد، مورد استفاده قرار گیرد.

## HKEY\_LOCAL\_MACHINE

زیرشاخه‌های این کلید، وضعیت فیزیکی کامپیوتر را مشخص می‌کنند که شامل حافظه‌ی سیستم، سخت‌افزار و نرم‌افزارهای نصب شده بر روی سیستم، اطلاعات پیکربندی، تنظیمات ورود به سیستم، اطلاعات امنیتی شبکه و اطلاعات دیگر سیستم است.

## HKEY\_USERS

زیرشاخه‌های این کلید، پیکربندی کاربران پیش فرض، جدید، جاری سیستم و به طور کلی همه‌ی کاربران را مشخص می‌کند.

## HKEY\_CURRENT\_CONFIG

زیرشاخه‌های این کلید، اطلاعاتی درباره وضعیت سخت‌افزار کامپیوتر در اختیار ما می‌گذارند. در واقع این کلید نام مستعاری برای کلید HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Hardware Profiles\Current است که در ویندوزهای قبل از NT ۳.۵۱ وجود نداشته است.

## کندوهای رجیستری

یک کندو (Hive) یک گروه از کلیدها، زیرکلیدها و مقادیر در رجیستری است که یک مجموعه از فایل‌های پشتیبان را به همراه دارد. در هنگام بوت ویندوز، اطلاعات از این فایل‌ها استخراج می‌شوند. شما هم چنین می‌توانید با استفاده از Import در منوی فایل registry editor به صورت دستی این کار را انجام دهید. زمانی که ویندوز را خاموش می‌کنید، اطلاعات کندوها در فایل‌های پشتیبان نوشته می‌شوند. شما می‌توانید این کار را به طور دستی با Export در منوی فایل registry editor نیز انجام دهید.

فایل‌های پشتیبان همه کندوها به جز HKEY\_CURRENT\_USER در شاخه‌ی Windows Root\System32\config قرار دارند. فایل‌های پشتیبان HKEY\_CURRENT\_USER در شاخه‌ی System Root\Documents and Settings\Username قرار دارند. پسوند فایل‌ها در این شاخه‌ها، نوع داده‌هایی که در بر دارند را نشان می‌دهند. در جدول زیر برخی کندوها و فایل‌های پشتیبانشان آمده است.

فایل‌های پشتیبان	کندوی رجیستری
System, System.alt, System.log, System.sav	HKEY_CURRENT_CONFIG
Ntuser.dat, Ntuser.dat.log	HKEY_CURRENT_USER
Sam, Sam.log, Sam.sav	HKEY_LOCAL_MACHINE\SAM
Security, Security.log, Security.sav	HKEY_LOCAL_MACHINE\Security
Software, Software.log, Software.sav	HKEY_LOCAL_MACHINE\Software
System, System.alt, System.log, System.sav	HKEY_LOCAL_MACHINE\System
Default, Default.log, Default.sav	HKEY_USERS\DEFAULT

هر زمان که یک کاربر به کامپیوتر وارد می‌شود، یک کندوی جدید با فایل‌های مجزا برای آن کاربر ساخته می‌شود که کندوی

پرو فایل کاربر نام دارد. یک کندوی کاربر، اطلاعاتی شامل تنظیمات برنامه‌های کاربر، تصویر زمینه، ارتباطات شبکه و پرینترها را در بر دارد. کندوهای پرو فایل کاربر در کلید HKEY\_USERS قرار دارند. مسیر فایل‌های پشتیبان این کندوها در کلید HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList\SID\ ProfileImagePath مشخص شده است. مقدار ProfileImagePath مسیر پرو فایل کاربر و نام کاربر را مشخص می‌کند.

## دسته بندی اطلاعات

قبل از قرار دادن اطلاعات در رجیستری باید آن‌ها را به دو دسته اطلاعات کامپیوتر و اطلاعات کاربر تقسیم کرد. با این تقسیم بندی، چندین کاربر می‌توانند از یک برنامه استفاده کنند و یا اطلاعات را بر روی شبکه قرار دهند. زمانی که یک برنامه نصب می‌شود، باید اطلاعات کامپیوتری خود را در شاخه فرضی HKEY\_LOCAL\_MACHINE\Software\MyCompany\MyProduct\1.0 به گونه‌ای تعریف کند که نام شرکت، نام محصول و نسخه برنامه به خوبی مشخص گردند و هم چنین اطلاعات مربوط به کاربران را در شاخه فرضی HKEY\_CURRENT\_USER\Software\MyCompany\MyProduct\1.0 نگاه دارد.

## باز کردن، ساختن و بستن کلیدها

قبل از آن که بتوانیم یک اطلاعات را در رجیستری درج کنیم، باید یک کلید بسازیم و یا یک کلید موجود را باز کنیم. یک برنامه همیشه به یک کلید به عنوان زیرکلیدی از یک کلید باز رجوع می‌کند. کلیدهای از پیش تعریف شده همیشه باز هستند. کلاس‌های تعریف شده برای کار با رجیستری در فضا نام Microsoft.Win32.Registry قرار دارند. کلاس Microsoft.Win32.Registry مربوط به کلاس‌های از پیش تعریف شده و کلاس Microsoft.Win32.RegistryKey برای کار با رجیستری است. برای باز کردن یک کلید از متد RegistryKey.OpenSubKey استفاده می‌کنیم. به یاد داشته باشید که کلیدهای از پیش تعریف شده همیشه باز هستند و نیازی به باز کردن ندارند. برای ساختن یک کلید از متد RegistryKey.CreateSubKey استفاده می‌کنیم. دقت کنید زیرکلیدی که می‌خواهید بسازید، باید به یک کلید باز رجوع کند. برای خاتمه دسترسی به یک کلید، باید آن را ببندیم. برای بستن یک کلید از متد RegistryKey.Close استفاده می‌کنیم.

اکنون که با ساختار رجیستری و کلاس‌های مربوطه در NET. برای کار با رجیستری آشنا شدیم، به کدنویسی می‌پردازیم.

## ساختن یک زیرکلید جدید

برای ساختن یک زیرکلید جدید از متد RegistryKey.CreateSubKey به صورت زیر استفاده می‌کنیم.

```
public RegistryKey CreateSubKey( string subkey);
```

subkey نام و مسیر کلیدی که می‌خواهید بسازید را مشخص می‌کند که معمولاً به فرم key name\Company Name\Application Name\version است. این متد یک زیرکلید را برمی‌گرداند و در صورت بروز خطا مقدار null را برمی‌گرداند و یک exception را فرا می‌خواند. خطا به دلایلی چون عدم داشتن مجوز، وجود نداشتن مسیر درخواستی و غیره رخ می‌دهد. برای بررسی exception ها می‌توانید از بلوک try-catch استفاده کنید.

```
RegistryKey MyReg = Registry .CurrentUser.CreateSubKey( "SOFTWARE\\SomeCompany\\SomeApp\\SomeVer" );
```

مثال فوق یک زیرکلید جدید در مسیر تعیین شده در شاخه‌ی HKEY\_CURRENT\_USER می‌سازد. برای دست یابی به کلیدهای از پیش تعریف شده از کلاس Registry مطابق جدول زیر استفاده می‌کنیم.

کلید	فیلد
HKEY_CLASSES_ROOT	ClassesRoot

کلید	فیلد
HKEY_CURRENT_USER	CurrentUser
HKEY_LOCAL_MACHINE	LocalMachine
HKEY_USERS	Users
HKEY_CURRENT_CONFIG	CurrentConfig

چند نکته حائز اهمیت است. اگر یک زیرکلید با نام مشابه در مسیر تعیین شده وجود داشته باشد، هیچ کلیدی ساخته نمی‌شود. حقیقت آن است که از متد `CreateSubKey` برای باز کردن یک کلید نیز می‌توانیم استفاده کنیم. متد `CreateSubKey` زیرکلید را همیشه در حالت ویرایش باز می‌گرداند. متد `CreateSubKey` دو پارامتر دیگر به عنوان ورودی دریافت می‌کند که از دو کلاس `RegistryKeyPermissionCheck` و `RegistryOptions` استفاده می‌کند. `RegistryKeyPermissionCheck` مشخص می‌کند که درخت زیرکلید، فقط خواندنی یا قابل ویرایش باشد. `RegistryOptions` مشخص می‌کند که اطلاعات کلید فقط در حافظه‌ی اصلی باشد و دیگر به کندوها منتقل نشود یعنی به طور موقتی باشد یا به طور پیش فرض دائمی باشد.

### باز کردن زیرکلید موجود

برای باز کردن یک زیرکلید موجود از متد `RegistryKey.OpenSubKey` به دو صورت استفاده می‌کنیم.

```
public RegistryKey OpenSubKey( string name);
public RegistryKey OpenSubKey( string name, bool writable);
```

صورت اول، کلید را در حالت فقط خواندنی باز می‌کند و صورت دوم، اگر `writable`، `true` باشد کلید را در حالت ویرایش باز می‌کند و اگر `false` باشد کلید را در حالت فقط خواندنی باز می‌کند. در هر دو صورت `name`، نام و مسیر زیرکلیدی که می‌خواهید باز کنید را مشخص می‌کند. اگر با خطا مواجه نشوید، متد زیرکلید را برمی‌گرداند، در غیر این صورت مقدار `null` را برمی‌گرداند.

```
RegistryKey MyReg = Registry.CurrentUser.OpenSubKey( "SOFTWARE\\SomeCompany\\SomeApp\\SomeVer" , true );
```

مثال فوق کلید مشخص شده را در شاخه‌ی `HKEY_CURRENT_USER` و در حالت ویرایش باز می‌کند.

### خواندن اطلاعات از رجیستری

اگر یک شیء `RegistryKey` سالم داشته باشید می‌توانید به مقادیر و اطلاعات درون مقادیر آن دسترسی داشته باشید. برای دست یابی به اطلاعات درون یک مقدار مشخص در کلید از متد `RegistryKey.GetValue` به دو صورت استفاده کنیم.

```
public object GetValue( string name);
public object GetValue( string name, object defaultValue);
```

صورت اول، اطلاعات درون مقداری با نام و مسیر `name` را برمی‌گرداند. اگر مقدار مذکور وجود نداشته باشد، مقدار `null` را برمی‌گرداند. در صورت دوم اگر مقدار خواسته شده وجود نداشته باشد، `defaultValue` را برمی‌گرداند. متد `GetValue` یک مقدار از نوع `object` را برمی‌گرداند در نتیجه شما برای استفاده، باید آن را به نوعی که می‌خواهید تبدیل کنید.

### نوشتن اطلاعات در رجیستری

برای نوشتن اطلاعات در یک مقدار از متد `RegistryKey.SetValue` به صورت زیر استفاده می‌کنیم.

```
public void SetValue( string name, object value);
```

رشته name ، نام مقداری که اطلاعات باید در آن ذخیره شود و value اطلاعاتی که باید در آن مقدار ذخیره شود را مشخص می‌کند. چون value از نوع object است می‌توانید هر مقداری را به آن بدهید. Value به طور اتوماتیک به DWORD یا باینری یا رشته‌ای تبدیل می‌شود. البته یک پارامتر سومی نیز وجود دارد که از کلاس RegistryValueKind استفاده کرده و نوع اطلاعات را به طور دقیق مشخص می‌کند. برای ذخیره اطلاعات در مقدار پیش فرض ( Default ) کافی است که مقدار name را برابر string.Empty قرار دهید. هر کلید می‌تواند یک مقدار پیش فرض داشته باشد که باید نام آن مقدار را Default قرار دهید.

### بستن یک کلید

زمانی که دیگر با کلید کاری ندارید و می‌خواهید تغییرات در رجیستری ثبت گردد باید فرآیندی به نام flushing را انجام دهید. برای انجام این کار به راحتی از متد RegistryKey.Close استفاده کنید.

```
RegistryKey MyReg = Registry.CurrentUser.CreateSubKey( "SOFTWARE\\SomeCompany\\SomeApp\\SomeVer" );
int nSomeVal = ( int )MyReg.GetValue( "SomeVal" , 0);
MyReg.SetValue( "SomeValue" , nSomeVal + 1);
MyReg.Close();
```

### پاک کردن یک کلید

برای پاک کردن یک زیرکلید از متد RegistryKey.DeleteSubKey به دو صورت استفاده می‌کنیم.

```
public void DeleteSubKey( string subkey);
public void DeleteSubKey( string subkey, bool throwOnMissingSubKey);
```

در صورت اول زیرکلید subkey را به شرطی حذف می‌کند که زیرکلید مذکور موجود باشد و زیرکلید دیگری در زیر آن نباشد. در صورت دوم نیز این شروط برقرار است با این تفاوت که اگر زیرکلید مذکور یافت نشود و throwOnMissingSubKey مقدار true داشته باشد یک exception فرا می‌خواند.

### پاک کردن کل یک درخت

برای پاک کردن کل یک درخت با تمامی کلیدهای فرزند و مقادیر آنها از متد RegistryKey.DeleteSubKeyTree به دو صورت استفاده می‌کنیم.

```
public void DeleteSubKeyTree( string subkey);
public void DeleteSubKeyTree( string subkey, bool throwOnMissingSubKey);
```

دیگر با پارامترهای ارسالی در این متد آشنایی دارید و لازم به توضیح نیست.

### پاک کردن یک مقدار

برای پاک کردن یک مقدار از متد RegistryKey.DeleteValue به دو صورت زیر استفاده می‌کنیم.

```
public void DeleteValue( string name);
public void DeleteValue( string name, bool throwOnMissingValue);
```

### لیست کردن زیرکلیدها

برای به دست آوردن یک لیست از همه زیرکلیدهای یک شیء RegistryKey از متد RegistryKey.GetSubKeyNames به صورت زیر

استفاده می‌کنیم که یک آرایه رشته‌ای از نام زیرکلیدها را برمی‌گرداند.

```
public string [] GetSubKeyNames();
```

هم چنین می‌توانید برای شمردن تعداد زیرکلیدها از خصوصیت `RegistryKey.SubKeyCount` استفاده نمایید.

### لیست کردن نام مقادیر

برای به دست آوردن یک لیست از همه مقادیری که در یک شیء `RegistryKey` وجود دارند از متد `RegistryKey.GetValueNames` به صورت زیر استفاده می‌کنیم که یک آرایه رشته‌ای از نام مقادیر را برمی‌گرداند.

```
public string [] GetSubKeyNames();
```

هم چنین می‌توانید برای شمردن تعداد زیرکلیدها از خصوصیت `RegistryKey.ValueCount` استفاده نمایید.

### ثبت تغییرات به صورت دستی

برای ثبت تغییرات یا به اصطلاح فلاش کردن به صورت دستی می‌توانید از متد `RegistryKey.Flush` به صورت زیر استفاده نمایید. زمانی که از `RegistryKey.Close` استفاده می‌کنید فرآیند فلاش کردن به طور اتوماتیک انجام می‌گیرد.

```
public void Flush();
```

## نظرات خوانندگان

نویسنده: سید ایوب کوکبی  
تاریخ: ۱۳۹۲/۰۹/۲۴ ۱۲:۳۳

ممنونم با وجود دات نت و معماری جدید برنامه ها، معمولا در چه شرایطی استفاده از رجیستری دلیل قانع کننده ای داره؟ و اون دلیل قانع کننده چیه؟

نویسنده: محسن خان  
تاریخ: ۱۳۹۲/۰۹/۲۴ ۱۲:۴۵

گاهی از اوقات برنامه های دسکتاپ نیاز پیدا می کنند تا به اطلاعات سیستمی دسترسی پیدا کنند یا آن ها را تغییر دهند. مثلا برنامه ای که نیاز داره در حین آغاز ویندوز، شروع به کار کنه، نیاز به ثبت خودش در رجیستری داره. یا اگر برنامه ای نیاز داشت که مثلا با Adobe reader کار کنه، می تونه مسیر دقیق نصب اون رو از رجیستری بخونه و از این موارد سیستمی زیاد هست.



عنوان:	SQL Instance
نویسنده:	م منفرد
تاریخ:	۱۹:۲۵ ۱۳۹۲/۱۰/۱۲
آدرس:	<a href="http://www.dotnettips.info">www.dotnettips.info</a>
گروه‌ها:	C#, SQL Server, Registry

ممکن است کاربر بر روی سیستم خود نسخه‌های مختلفی از SQL Server را نصب کرده باشد. برای مثال SQL Express, SQL 2005, SQL 2008. و یا نسخه ای خاص (مثلا 2012) را چند بار روی سیستم خود نصب کرده باشد. SQL برای تفکیک این نسخه‌ها و نصب‌ها از مفهومی با عنوان Instance استفاده می‌کند. یعنی به هر نسخه نصب شده نامی یکتا می‌دهد تا بتوان به تفکیک به آنها دسترسی داشت.

برای اتصال به این نسخه‌ها باید در بخش آدرس سرور، از ترکیب نام سیستم و نام Instance به این شکل استفاده کرد:  
SystemName\Instance

بعضی مواقع لازم است که لیست Instance‌های نصب شده روی سیستم کاربر را به دست آوریم. ADO.NET کلاسی به همین منظور تعبیه کرده که شبکه را جستجو کرده و SQL Instance‌های مختلف را که قابل دسترسی هستند را برای شما لیست می‌کند. استفاده از این کلاس بسیار ساده است:

```
using System.Data.Sql;

class Program
{
    static void Main()
    {
        // Retrieve the enumerator instance and then the data.
        SqlDataSourceEnumerator instance =
            SqlDataSourceEnumerator.Instance;
        System.Data.DataTable table = instance.GetDataSources();

        // Display the contents of the table.
        DisplayData(table);

        Console.WriteLine("Press any key to continue.");
        Console.ReadKey();
    }

    private static void DisplayData(System.Data.DataTable table)
    {
        foreach (System.Data.DataRow row in table.Rows)
        {
            foreach (System.Data.DataColumn col in table.Columns)
            {
                Console.WriteLine("{0} = {1}", col.ColumnName, row[col]);
            }
            Console.WriteLine("=====");
        }
    }
}
```

البته با توجه به اینکه شبکه را جستجو می‌کند در نرم افزار شما وقفه خواهد انداخت. خوب اگه بخواهیم Instance‌های نصب شده روی سیستم کاربر را پیدا کنیم چی؟ ساده‌ترین و سریعترین راه استفاده از رجیستری سیستم است. نام Instance‌ها در رجیستری ویندوز در آدرس زیر قابل دسترسی است:

HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\Instance Names

برای استفاده از این کلید در #c می‌توان از کد زیر کمک بگیرید:

```
var key = Registry.LocalMachine.OpenSubKey(@"SOFTWARE\Microsoft\Microsoft SQL Server\Instance Names");

foreach (string sk in key.GetSubKeyNames())
{
    var rkey = key.OpenSubKey(sk);
    foreach (string s in rkey.GetValueNames())
    {
        MessageBox.Show("Sql instance name:" + s);
    }
}
```

فقط دو نکته قابل توجه است. برنامه باید در Any CPU کامپایل شود تا در سیستم‌های 64 بیتی بتوانید به محل درست رجیستری دسترسی پیدا کنید. چون نرم افزارهای 32 بیت در ویندوز 64 بیت در سیستم wow64 اجرا می‌شود که دسترسی به رجیستری آن در آدرس wow64 هر قسمت رجیستری است. بنابراین کد فوق در حالت Any CPU و غیر فعال بودن Prefer 32-bit قسمت Build در Properties برنامه به درستی اجرا می‌شود.

نکته: Default Instance در SQL مقدار MSSQLSERVER می‌باشد.