

Программирование реестра средствами .NET на языке C# .

Автор : Котов Александр Евгеньевич aka Black_c0der .

Что такое реестр ?

Реестр действует как центральное хранилище сведений об операционной системе и приложений компьютера. Например, приложения используют реестр для хранения сведений, которые необходимо сохранить при закрытии приложения и к которым при перезагрузке приложения необходимо получить доступ. Например, имеется возможность сохранять цветовые настройки, положение или размер окна. Каждый пользователь имеет возможность сохранять эти сведения для себя в определенном месте реестра.

Основные классы для работы с реестром в .NET – Registry и RegistryKey , которые находятся в пространстве имён Microsoft.Win32 .

Registry содержит открытые (статические) поля которые предоставляют доступ к соотв. Ключам реестра :

LocalMachine – Данные о конфигурации для локального компьютера. Это поле считывает базовый ключ реестра HKEY_LOCAL_MACHINE.

ClassesRoot – Определение типов (или классов) документов и свойств, связанных с этими типами. Это поле считывает базовый ключ реестра Windows HKEY_CLASSES_ROOT.

CurrentConfig – Сведения о конфигурации, относящиеся к оборудованию, которое не имеет прямого отношения к пользователю. Это поле считывает базовый ключ реестра HKEY_CURRENT_CONFIG.

Users – Сведения о стандартной пользовательской конфигурации. Это поле считывает базовый ключ реестра HKEY_USERS.

CurrentUser – Сведения о текущих пользовательских привилегиях. Это поле считывает базовый ключ реестра HKEY_CURRENT_USER.

PerformanceData – Сведения о рабочих характеристиках для программных компонентов. Это поле считывает базовый ключ реестра HKEY_PERFORMANCE_DATA.

DynData – Динамические данные реестра. Это поле считывает базовый ключ реестра HKEY_DYN_DATA. (Windows 9x/ME) .

Теперь рассмотрим класс **RegistryKey** , который является базовой единицей организации в реестре и может быть сравнен с папками в Windows Explorer.

Основные свойства RegistryKey :

Name	Извлечение имени ключа. .
SubKeyCount	Извлечение числа подключей на базовом уровне для текущего ключа.
ValueCount	Извлечение числа значений в ключе.

Методы :

Close	Закрытие ключа и его сброс на диск, если изменено содержание.
CreateSubKey	Создание нового подключа или открытие существующего подключа. Строка subKey не учитывается в регистре.
DeleteSubKey	Перегружен. Удаление заданного подключа. Строка subKey не учитывается в регистре.

DeleteSubKeyTree	Удаление подключа и дочерних подключей рекурсивно. Строка subKey не учитывается в регистре.
DeleteValue	Перегружен. Удаление заданного значения из этого ключа.
Flush	Запись всех атрибутов заданного открытого ключа реестра в реестр.
GetSubKeyNames	Извлечение массива строк, который содержит все имена подключей.
GetValue	Перегружен. Извлечение заданного значения.
GetValueNames	Извлечение массива строк, содержащего все имена значений, связанных с этим ключом.
OpenRemoteBaseKey	Открытие нового RegistryKey, который представляет запрошенный ключ на чужом компьютере.
OpenSubKey	Перегружен. Извлечение заданного подключа с доступом для записи.
SetValue	Установка заданного значения.

Ну вот , всё что нужно знать перечислил , теперь можно попробовать что-нибудь записать в реестр и прочитать из него .

Допустим , мы хотим чтобы при выходе из программы сохранялся размер окна (формы) , а при запуске программы , окно становилось таким , какое было перед закрытием .

В деструктор формы пишем код :

```
protected override void Dispose(bool disposing)
{
    /////////////////////////////////// Вот это добавили ///////////////////////////////////
    try
    {
        RegistryKey key = Registry.CurrentUser.OpenSubKey("Software", true);
        // Открыли папку , true означает - хотим ли мы записывать в этот раздел
        // реестра ?
        RegistryKey wKey = key.CreateSubKey("RegistryTesting");
        // Создали новую папку в реестре
        wKey.SetValue("FormWidth", this.Size.Width);
        wKey.SetValue("FormHeight", this.Size.Height);
        // Здесь мы создали 2 ключа в которых сохранили размер формы
        // MessageBox.Show("Параметры сохранены .");

    }
    catch(System.Exception err)
    {
        MessageBox.Show("Произошла ошибка при сохранении параметров : " +
            err.Message);
    }

    ///////////////////////////////////
    if (disposing && (components != null))
    {
        components.Dispose();
    }

    base.Dispose(disposing);
}
```

***Примечание :** Если вы пишете свою программу в Visual Studio 2005 , тогда не забудьте добавить в модуль с деструктором (Имя_Формы.Designer.cs) эти директивы :

```
using Microsoft.Win32;  
using System.Windows.Forms;
```

Ну вот параметры формы сохранились , в этом легко убедиться если запустить regedit и посмотреть раздел HKEY_CURRENT_USER\Software\RegistryTesting .

Теперь сделаем так чтобы форма восстанавливала свои размеры при загрузке . В событие формы OnLoad пишем такой код :

```
private void frmMain_Load(object sender, EventArgs e)  
{  
    try  
    {  
        RegistryKey key = Registry.CurrentUser;  
        key = key.OpenSubKey("Software\\RegistryTesting");  
        System.Object w = key.GetValue("FormWidth");  
        System.Object h = key.GetValue("FormHeight");  
        // Получили значения ключей и теперь применяем их к форме  
        this.Width = (int)w;  
        this.Height = (int)h;  
        // MessageBox.Show("Форма восстановлена .");  
    }  
    catch(System.Exception err)  
    {  
        MessageBox.Show("Произошла ошибка при загрузке параметров  
        "+err.Message);  
    }  
}
```

Теперь можно запустить программу и посмотреть как это работает .

Чтобы удалить все параметры из реестра , достаточно удалить папку RegistryTesting вызвав метод DeleteSubKey :

```
RegistryKey key = Registry.CurrentUser.OpenSubKey("Software", true);  
key.DeleteSubKey("RegistryTesting");
```

Ну вот и всё, теперь вы можете писать свои программы по работе с реестром , пользуясь основными свойствами и методами классов Registry и RegistryKey . Надеюсь эта статья не оказалась напрасной .