Работа с реестром и файловой системой

Что такое реестр

Реестр в операционной системе Windows - это иерархическая база данных, которая хранит настройки и конфигурационные данные операционной системы, установленных на ней приложений и компонентов.

Реестр представляет собой иерархию ключей и подключей, которые можно рассматривать как папки и подпапки в файловой системе.

Зачем он нужен

Реестр содержит сведения, на которые Windows постоянно ссылается во время операции, такие как профили для каждого пользователя, приложения, установленные на компьютере, типы документов, которые могут создаваться, параметры таблицы свойств для папок и значков приложений, оборудование, которое установлено в системе, и используемые порты.

Ветви реестра

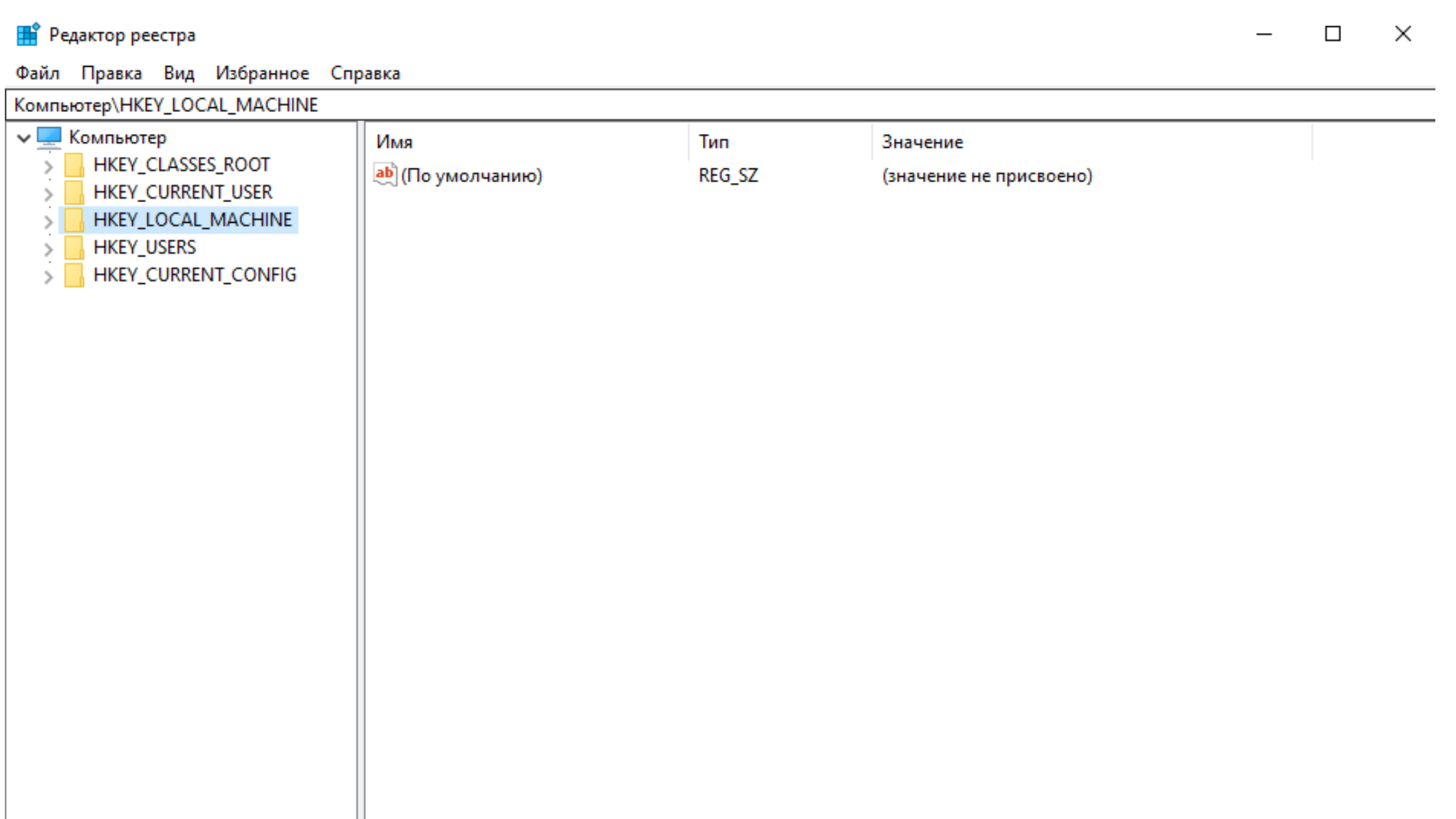
- HKEY_LOCAL_MACHINE
- HKEY_CURRENT_USER
- HKEY_CLASSES_ROOT
- HKEY_USERS
- HKEY_CURRENT_CONFIG

Ключи

Параметры или ключи реестра имеют имена, представленные в обычном текстовом виде и значения, которые хранятся в виде стандартизированных записей определенного типа

Работа с реестром

Для просмотра реестра мы можем воспользоваться утилитой regedit. При запуске данная утилита отображает в графическом окне в древовидном виде все содержимое реестра:


Редактор реестра
Файл Правка Вид Избранное Справка
Компьютер\HKEY_LOCAL_MACHINE
Компьютер
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS
HKEY_CURRENT_CONFIG
Имя
Тип
Значение
(По умолчанию)
REG_SZ
(значение не присвоено)

Работа с реестром на C#

Для управления регистром в пространстве имен Microsoft.Win32 имеются два класса: Registry и RegistryKey.

Класс Registry позволяет получить доступ к ключам верхнего уровня реестра. А класс RegistryKey представляет отдельный ключ реестра.

Registry

Класс Registry содержит статические свойства, каждое из которых представляет соответствующий ключ верхнего уровня:

- Registry.ClassesRoot
- Registry.CurrentConfig
- Registry.CurrentUser

Registry

- Registry.DynData
- Registry.Users
- Registry.PerformanceData

RegistryKey

Для управления ключами в реестре класс RegistryKey определяет ряд свойств и методов. Основные из них:

- Name: возвращает имя ключа реестра
- Close(): закрывает ключ
- CreateSubKey(): создает вложенный ключ, если он не существует
- DeleteSubKey(): удаляет вложенный ключ

RegistryKey

- DeleteValue(): удаляет значение ключа
- GetSubKeyNames(): возвращает коллекцию имен вложенных ключей
- GetValue(): возвращает значение ключа
- OpenSubKey(): открывает вложенный ключ
- SetValue(): устанавливает значение ключа

Пример

```
RegistryKey currentUserKey = Registry.CurrentUser;
RegistryKey helloKey =
currentUserKey.CreateSubKey("HelloKey");
helloKey.SetValue("login", "admin");
helloKey.SetValue("password", "12345");
helloKey.Close();
```

Пример

```
RegistryKey currentUserKey = Registry.CurrentUser;
RegistryKey helloKey = currentUserKey.OpenSubKey("HelloKey", true);
RegistryKey subHelloKey = helloKey.CreateSubKey("SubHelloKey");
subHelloKey.SetValue("val", "23");
subHelloKey.Close();
helloKey.Close();
```

Пример

```
RegistryKey currentUserKey = Registry.CurrentUser;
RegistryKey helloKey = currentUserKey.OpenSubKey("HelloKey");
string login = helloKey.GetValue("login").ToString();
string password = helloKey.GetValue("password").ToString();
helloKey.Close();
Console.WriteLine(login);
 Console.WriteLine(password);
```

Пример

```
RegistryKey currentUserKey = Registry.CurrentUser;
RegistryKey helloKey = currentUserKey.OpenSubKey("HelloKey", true);
helloKey.DeleteSubKey("SubHelloKey"); // удаляем вложенный ключ
helloKey.DeleteValue("login");// удаляем значение из ключа
helloKey.Close();
currentUserKey.DeleteSubKey("HelloKey");// удаляем сам ключ
```

Задание

Добавить блокнот в автозагрузку. Открыть раздел HKLM\Software\Microsoft\Windows\CurrentVersion. Найти подраздел Run. В этом разделе есть строковые ключи , отвечающие за запуск программ. Название ключа может быть произвольным, а в качестве значения у них указывается запускаемая программа, если надо - то с параметрами.

Сжатие данных

Сжатие файла — это уменьшение его размера при сохранении исходных данных. В этом случае файл занимает меньше места на устройстве, что также облегчает его хранение и передачу через интернет или другим способом. Важно отметить, что сжатие не безгранично и обычно делится на два основных типа: с потерями и без потерь.

Сжатие с потерями

Такой способ уменьшает размер файла, удаляя ненужные биты информации. Чаще всего встречается в форматах изображений, видео и аудио, где нет необходимости в идеальном представлении исходного медиа. MP3 и JPEG — два популярных примера.

Сжатие без потерь

Сжатие без потерь позволяет уменьшить размер файла так, чтобы в дальнейшем можно было восстановить первоначальное качество. В отличие от сжатия с потерями, этот способ не удаляет никакую информацию.

Сжатие данных

.NET предоставляет классы, которые позволяют сжимать файлы, а также затем восстанавливать их в исходное состояние.

Это классы ZipFile, DeflateStream и GZipStream, которые находятся в пространстве имен System.IO.Compression и представляют реализацию одного из алгоритмов сжатия Deflate или GZip.

ZipFile

Класс ZipFile позволяет создавать архив из каталогов. Его основные методы:

void CreateFromDirectory(string sourceDirectoryName, string destinationFileName): архивирует папку по пути sourceDirectoryName в файл с названием destinationFileName

ZipFile

void ExtractToDirectory(string sourceFileName, string destinationDirectoryName): извлекает все файлы из zip-файла sourceFileName в каталог destinationDirectoryName

ZipFile

```
string sourceFolder = "D://test/"; // исходная папка
string zipFile = "D://test.zip"; // сжатый файл
string targetFolder = "D://newtest"; // папка, куда распаковывается файл
ZipFile.CreateFromDirectory(sourceFolder, zipFile);
Console.WriteLine($"Папка {sourceFolder} архивирована в файл {zipFile}");
ZipFile.ExtractToDirectory(zipFile, targetFolder);
Console.WriteLine($"Файл {zipFile} распакован в папку {targetFolder}");
```

GZipStream

GZipStream(Stream stream, CompressionLevel level): stream представляет данные, а level задает уровень сжатия

GZipStream(Stream stream, CompressionMode mode): mode указывает, будут ли данные сжиматься или, наоборот, восстанавливаться и может принимать два значения:

- CompressionMode.Compress: данные сжимаются

-CompressionMode.Decompress: данные восстанавливаются