

# Программирование ввода-вывода

**Курс:** C# Professional

**Средства обучения:** Компьютер с установленной Visual Studio 2012

## Обзор, цель и назначение урока

На этом уроке рассматривается работа со средствами ввода-вывода в Microsoft .Net Framework. Система ввода-вывода предоставляет средства доступа к файлам и папкам файловой системы, чтения/записи, сжатия потоков и механизмов изолированного хранения.

## Изучив материал данного занятия, учащийся сможет:

- Использовать классы [File](#) и [FileInfo](#)
- Использовать классы [Directory](#) и [DirectoryInfo](#)
- Использовать классы [DriveInfo](#) и [DriveType](#)
- Перечислять файлы, каталоги и диски, используя классы, производные от класса [FileSystemInfo](#)
- Использовать класс [Path](#) для управления путями файловой системы
- Отслеживать изменения в файловой системе, используя класс [FileSystemWatcher](#)
- Открывать файлы, читать их содержимое, совершать запись в файлы
- Создавать потоки (streams) в памяти
- Сжимать потоки при помощи классов [GZipStream](#) и [DeflateStream](#)
- Производить декомпрессию потоков при помощи классов [GZipStream](#) и [DeflateStream](#)
- При помощи класса [IsolatedStorageFile](#) получать доступ к изолированному хранилищу для сохранения данных программы
- Создавать файлы и папки в изолированном хранилище, используя класс [IsolatedStorageFileStream](#)
- Получать доступ к различным областям внутри изолированного хранилища, специфичным для пользователя и компьютера, используя класс [IsolatedStorageFile](#)

## Содержание урока

1. Навигация по файловой системе
2. Чтение и запись файлов
3. Работа с потоками
4. Работа с изолированным хранилищем

## Резюме

- Для перечисления объектов файловой системы и получения подробной информации об их свойствах можно использовать классы [FileInfo](#), [DirectoryInfo](#) и [DriveInfo](#).
- Класс [Path](#) позволяет получать подробную информацию о путях файловой системы, его следует использовать вместо ручного разбора путей.
- Для отслеживания изменений файловой системы, таких как добавление, удаление и переименование файлов и папок, можно использовать класс [FileSystemWatcher](#).
- Класс [File](#) позволяет открывать, создавать, читать и записывать файлы целиком либо по частям.
- Класс [FileStream](#) представляет файл и позволяет выполнять чтение и запись.
- Чтобы упростить чтение-запись строк в потоки, используются классы [StreamReader](#) и [StreamWriter](#).
- Класс [MemoryStream](#) – специализированный поток, поддерживающий создание в памяти буфера чтения-записи и запись данных буферизованного потока в другие потоки.
- Классы потоков сжатия ([GZipStream](#) и [DeflateStream](#)) поддерживают сжатие-декомпрессию данных объемом до 4-х Гб.

- Классы потоков сжатия служат оболочками потоков, хранящих сжатые данные.
- Изолированное хранилище – это защищенная область для хранения данных, специфичных для сборки, пользователя или приложения. Для работы с изолированным хранилищем не требуется высоких привилегий, поэтому приложения смогут хранить в нем свои данные, даже не обладая разрешениями на доступ к системе пользователя.
- Для хранения данных сборок и пользователей в защищенных областях используется класс `IsolatedStorageFile`.
- Класс `IsolatedStorageFileStream` позволяет обмениваться данными с безопасными хранилищами.
- Поскольку класс `IsolatedStorageFileStream` – потомок `FileStream`, с созданными им файлами можно работать, как с любыми другими файлами файловой системы.
- Класс `IsolatedStorageFilePermission` гарантирует наличие у кода разрешений, необходимых для взаимодействия с изолированным хранилищем.

### Закрепление материала

- Как правильно открыть файл для записи?
- Какие изменения можно отслеживать посредством `FileSystemWatcher`?
- Какие методы класса `FileStream` изменяют свойство `Position`?
- Как создать экземпляр класса `FileStream`?
- Можно ли сжимать данные, размеры которых превышают 4 Гб, при помощи классов `GZipStream` и `DeflateStream`?
- Какие методы используются для создания объектов `IsolatedStorageFile`?

### Дополнительное задание

Создайте на диске 100 директорий с именами от `Folder_0` до `Folder_99`, затем удалите их.

### Самостоятельная деятельность учащегося

#### Задание 1

Выучите основные конструкции, классы и понятия, рассмотренные на уроке.

#### Задание 2

Создайте файл, запишите в него произвольные данные и закройте файл. Затем снова откройте этот файл, прочитайте из него данные и выведите их на консоль.

#### Задание 3

Напишите приложение для поиска заданного файла на диске. Добавьте код, использующий класс `FileStream` и позволяющий просматривать файл в текстовом окне. В заключение добавьте возможность сжатия найденного файла.

#### Задание 4

Создайте приложение WPF Application, позволяющее пользователям сохранять данные в изолированное хранилище.

#### Задание 5

Зайдите на сайт MSDN.

Используя поисковые механизмы MSDN, найдите самостоятельно описание темы по каждому примеру, который был рассмотрен на уроке, так, как это представлено ниже, в разделе «Рекомендуемые ресурсы», описания данного урока. Сохраните ссылки и дайте им короткое описание.

## Рекомендуемые ресурсы

MSDN: пространство имен System.IO

<http://msdn.microsoft.com/ru-ru/library/gg145019.aspx>

MSDN: класс `DirectoryInfo`

<http://msdn.microsoft.com/ru-ru/library/system.io.directoryinfo.aspx>

MSDN: класс `Path`

<http://msdn.microsoft.com/ru-ru/library/system.io.path.aspx>

MSDN: `IsolatedStorage` – Изолированное хранилище

<http://msdn.microsoft.com/ru-ru/library/3ak841sy.aspx>