Файловый и потоковый ввод-вывод

Введение

- Пространство имен System.IO содержит типы, позволяющие осуществлять чтение и запись в файлы и потоки данных, а также типы для базовой поддержки файлов и папок.
- Пространство имен System.IO содержит типы, позволяющие выполнять синхронное и асинхронное чтение и запись данных в потоки или файлы.
- Пространство имен System.IO предоставляет несколько классов, которые позволяют выполнять с файлами, каталогами и потоками различные действия, такие как чтение и запись.

Введение

- Файл — это именованная и упорядоченная коллекция отдельных последовательностей байтов, имеющих постоянное место хранения.
- При работе с файлами оперируют такими понятиями, как путь к каталогу, дисковый накопитель, имя файла и имя каталога.

Введение

- В отличие от файлов потоки предоставляют возможность писать и читать байты из вспомогательного запоминающего устройства, которым может являться одно из нескольких устройств хранения информации.
- Существуют как вспомогательные запоминающие устройства, отличные от дисков, так и несколько типов потоков, отличных от файловых потоков. Например, есть сетевой поток, поток памяти и поток для чтения и записи на магнитную ленту.

Понятие потока

- Поток — это абстракция последовательности байтов, например файл, устройство ввода-вывода, канал взаимодействия процессов или сокет TCP/IP.

Понятие потока

- Потоки включают три основные операции:
  - Чтение из потока — это перенос информации из потока в структуру данных, такую как массив байтов.
  - Запись в поток— это передача данных из структуры данных в поток.
  - Потоки поддерживают поиск. Поиск — это выяснение и изменение текущей позиции внутри потока. Возможность поиска зависит от вида резервного хранилища потока. Например, в сетевых потоках отсутствует унифицированное представление текущего положения, поэтому обычно они не поддерживают поиск.

Класс Stream

- Класс Stream является абстрактным базовым классом всех потоков.
- Абстрактный базовый класс Stream поддерживает чтение и запись байтов.
- Класс **Stream** поддерживает асинхронный ввод и вывод. По умолчанию его реализация определяет операции синхронного чтения и записи на основе соответствующих им асинхронных методов, и наоборот.

Класс Stream

- Все классы, которые работают с потоками, являются производными от класса **Stream**. Класс **Stream** и его производные классы предоставляют способ просмотра источников данных и хранилищ объектов, изолируя программиста конкретных от специфических деталей операционной системы и базовых устройств.

Терминология ввода-вывода

- При синхронным вводе-выводе функция ввода-вывода (Write, Read,) возвратит управление только после того, как полностью выполнится операция ввода-вывода. При использовании носителей с медленной скоростью чтения-записи, или когда производится обработка больших объёмов данных, то программа «подвиснет» на время чтения-записи.
- При асинхронном вводе-выводе функция сразу же возвращает управление, и программа продолжает выполняться дальше без задержек. Эта технология может пригодиться для разработки программ для взаимодействия с внешними устройствами с низкой скоростью передачи данных, например сотовыми телефонами, устройствами Bluetooth или IrDA.

Терминология ввода-вывода

- Синхронный ввод и вывод означает, что метод блокируется до тех пор, пока операция ввода или вывода не будет завершена.

Распространенные задачи с файлами

- Создание текстового файла.
- Запись в текстовый файл.
- Чтение из текстового файла.
- Добавление текста в файл.
- Переименование или перемещение файла.
- Удаление файла.
- Копирование файла.
- Получение сведений о размере файла.

Распространенные задачи с файлами

- Получение атрибутов файла.
- Установка атрибутов файла.
- Определение существования файла.
- Чтение из двоичного файла.
- Запись в двоичный файл.
- Извлечение расширения файла.
- Извлечение полного пути к файлу.
- Извлечение имени и расширения файла из его пути.
- Изменение расширения файла.

Распространенные задачи с каталогами

- Переименование или перемещение каталога.
- Копирование каталога.
- Удаление каталога.
- Создание каталога.
- Создание вложенного каталога.
- Просмотр файлов каталога.
- Просмотр вложенных каталогов в каталоге.
- Отображение всех файлов во всех вложенных каталогах в указанном каталоге.
- Определение размера каталога.
- Определение существования каталога.

Классы, используемые в файловом вводе и выводе

- **Directory** предоставляет статические методы операций создания, перемещения и перечисления в директориях и поддиректориях.
- **DirectoryInfo** предоставляет методы экземпляра операций создания, перемещения и перечисления в директориях и поддиректориях.
- **DriveInfo** предоставляет методы экземпляра для доступа к сведениям о диске.
- **File** предоставляет статические методы для создания, копирования, удаления, перемещения и открытия файлов, а также помогает при создании объектов FileStream.

Классы, используемые в файловом вводе и выводе

- **FileInfo** предоставляет методы экземпляра для создания, копирования, удаления, перемещения и открытия файлов, а также помогает при создании объектов FileStream.
- **FileStream** поддерживает произвольный доступ к файлам с помощью метода Seek.По умолчанию класс **FileStream** открывает файлы синхронно, но поддерживает и асинхронные операции.
- **FileSystemInfo** является абстрактным базовым классом для **FileInfo** и **DirectoryInfo**.
- **Path** предоставляет методы и свойства для обработки строк каталогов межплатформенным способом.

Классы, используемые в файловом вводе и выводе

- **DeflateStream** предоставляет методы и свойства для сжатия и распаковки потоков с использованием Deflate алгоритма.
- **GZipStream** предоставляет методы и свойства для сжатия и распаковки потоков. По умолчанию этот класс использует тот же алгоритм, что и класс DeflateStream, но он не может быть расширен для использования других форматов сжатия.
- **SerialPort** предоставляет методы и свойства для управления файлом ресурсов порта с последовательным выводом данных.
- Класс **File**, **FileInfo**, **DriveInfo**, **Path**, **Directory,** и **DirectoryInfo** являются изолированными.Можно создавать новые экземпляры этих классов, но они не могут иметь производных классов.

Классы, используемые для чтения и записи в поток

- Классы **BinaryReader** и **BinaryWriter** производят чтение и запись кодированных строк и простых типов данных в **потоки**.
- Класс **StreamReader** считывает символы из **потоков**, используя **Encoding** для преобразования символов в байты, и наоборот. **StreamReader** имеет конструктор, который пытается выяснить подходящую **кодировку** для заданного **потока**, на основе наличия специфичной для **кодировки** преамбулы, такой как метка порядка байтов.
- **StreamWriter** записывает символы в **потоки**, используя **кодировку** для преобразования символов в байты.

Классы, используемые для чтения и записи в поток

- **StringReader** считывает символы из **строк**. **StringReader** позволяет интерпретировать класс **Strings** с одним и тем же API-интерфейсом, поэтому в качестве выходных данных может использоваться класс **Stream** в любой кодировке, либо класс **String**.
- **StringWriter** записывает символы в **строки**. **StringWriter** позволяет интерпретировать класс **Strings** с одним и тем же API-интерфейсом, поэтому в качестве выходных данных может использоваться класс **Stream** в любой кодировке, либо класс **String**.
- **TextReader** является абстрактным базовым классом для класса **StringReader** и класса **StreamReader**. В то время как реализации абстрактного класса **Stream** предназначены для побайтового ввода и вывода, реализации **TextReader** предназначены для вывода знаков в кодировке Unicode.
- **TextWriter** является абстрактным базовым классом для класса **StringWriter** и класса **StreamWriter**. В то время как реализации абстрактного класса **Stream** предназначены для побайтового ввода и вывода, реализации **TextWriter** предназначены для ввода знаков в кодировке Unicode.