Разработка .NET приложений

Лекция 8

Работа с файлами

Сегодня

- Работа с файлами
  - Работа с файловой системой
  - Операции с потоками (**Stream**)

Работа с файлами

- Пространство имен **System.IO**
- Работа с файловой системой
  - Диски: *DriveInfo*
  - Папки: *DirectoryInfo*, *Directory*, *FileSystemInfo*
  - Файлы: *File*, *FileInfo*, *FileSystemInfo*
- Работа с путями
  - *Path*
- Наблюдение за изменениями в файловой системе
  - *FileSystemWatcher*
- Работа с потоками (чтение/запись файла)
  - *Stream*, производные от него и классы обертки

Файлы и папки

- Два способа работы:
  - **File, Directory** – статические классы
  - **FileInfo, DirectoryInfo**
  - Многие методы похожие
- **FileSystemInfo**
  - Базовый для **FileInfo** и **DirectoryInfo**
- **DriveInfo**
  - Информация о диске
  - Не связан с **FileSystemInfo**

DriveInfo

- Экземпляр представляет один диск
  - DriveInfo drive = new DriveInfo("c");
  - DriveInfo drive = new DriveInfo(@"\\server\share");
- Информация о диске системы
  - Name – имя диска
  - DriveType – тип диска. Например: Fixed, CDRom
  - IsReady – готов ли к использованию (например для CD Rom)
  - VolumeLabel – метка тома
  - DriveFormat – тип файловой системы
  - TotalSize – размер диска
  - AvailableFreeSpace – размер свободного места
  - …
- Получение всех дисков
  - Статически метод GetDrives()
    - DriveInfo[] drives = DriveInfo.GetDrives();

FileSystemInfo

- Абстрактный класс
- Базовый класс для **DirectoryInfo** и **FileInfo**
- Свойства файла/папки
  - **Name** – название файла, папки
  - **FullName** – полное имя, т.е. с полным путем
  - **Extension** - расширение
  - **Exists()** – существует ли такой файл / папка
  - **CreationTime**, **LastAccessTime** – время создания / последнего доступа (изменения)
  - **Attributes** - атрибуты (только на чтение)
  - ...

DirectoryInfo (и Directory)

- Экземпляр DirectoryInfo представляет одну папку файловой системы
- Наследник от FileSystemInfo
- Создание объекта (но не физической папки на диске)
  - DirectoryInfo windowsFolder = new DirectoryInfo(@"c:\Windows");
- Получение информации о файлах и папках
  - GetDirectories() – получение подпапок. Возможен поиск по шаблону и поиск во всех дочерних папках
    - DirectoryInfo[] folders = di.GetDirectories();
    - DirectoryInfo[] folders = di.GetDirectories(“my*”);
    - DirectoryInfo[] folders = di.GetDirectories(“my*”, SearchOption.AllDirectories);
  - GetFiles() – получение файлов
    - FileInfo[] files = folder.GetFiles();
    - FileInfo[] files = folder.GetFiles("*.txt”, SearchOption.TopDirectoryOnly);
  - GetFileSystemInfos() – получение всех файлов/папок
    - FileSystemInfo[] fsi = folder.GetFileSystemInfos();
- Операции с папками
  - Create() – создание папки по текущему объекту DirectoryInfo
    - new DirectoryInfo(@"c:\MyDir").Create();
  - CreateSubdirectory() - создание подпапки текущего каталога
  - Delete() – удаление текущей папки
  - MoveTo () – перемещение текущей папки
- Класс Directory имеет практически такие же методы, но в статическом исполнении
  - Directory.CreateDirectory(path);

FileInfo (и File)

- Экземпляр FileInfo представляет один файл файловой системы
- Наследник от **FileSystemInfo**
- Создание объекта (но не физического файла на диске)
  - FileInfo file = new FileInfo("c:\\test.txt");
- Свойства (в добавление к свойствам **FileSystemInfo**)
  - Directory, DirectoryName – папка **DirectoryInfo** или полное имя папки (с полным путем)
  - IsReadOnly – файл только для чтения
  - Length – размер файла в байтах
- Операции с файлами целиком
  - Create(), CreateText() – создание файла
  - CopyTo() – копирование файла
  - Delete() – удаление текущего файла
  - MoveTo () – перемещение текущего файла
- Операции с содержимым файла
  - Create(), CreateText(), OpenRead(), OpenText(), OpenWrite(), AppendText()
- Класс File имеет практически такие же методы, но в статическом исполнении
  - File.Delete("c:\\myText.txt");

Работа с путями Path

- Статический класс
- Работа с путями
  - GetPathRoot, GetDirectoryName, GetFullPath, GetFileName, GetFileNameWithoutExtension, GetExtension
- Комбинирование путей
  - Combine, ChangeExtension
- Проверки
  - GetInvalidPathChars, GetInvalidFileNameChars, IsPathRooted, HasExtension
- Работа с временными папками
  - GetTempPath , GetTempFileName

- Не изменяет сами объекты, т.е. не переименовывает и не перемещает (исключение **GetTempFileName -** создаёт физически файл на диске)
- Не проверяет наличие объектов физически*
- Проверяет валидность задания путей. Например проверяет допустимость символов

- Полезным может быть также статический класс **Environment**

FileSystemWatcher

- Слушает сообщения об изменениях в файловой системе и генерирует события
- Настройка:
  - Path – папка за изменениями которой нужно следить
  - Filter – задает фильтр объектов за которыми следить
  - NotifyFilter – задает за какими изменениями следить
    - (Attributes, CreationTime, DirectoryName, FileName, LastAccess, LastWrite, Security, Size)
  - Начало слежения – установка свойства EnableRaisingEvents = true
- Генерирует события
  - Changed, Created, Deleted, Renamed

Демонстрации

Работа с файлами каталогами

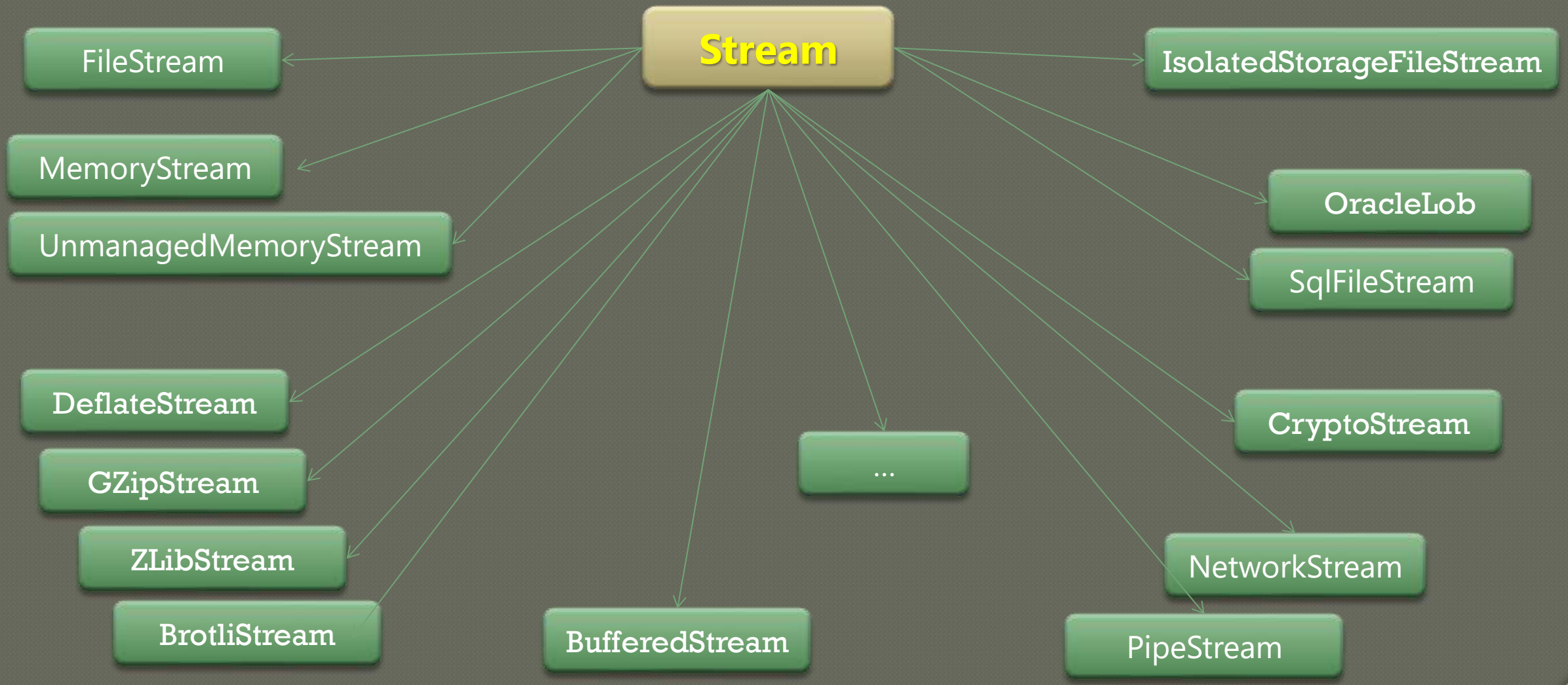
Ввод-вывод в .NET

- Базовый абстрактный класс для всех потоков – класс **Stream**
  - Представляет файл, порт, область памяти и т.д. как поток байт
  - Абстрактный класс
  - Запись, чтение, перемещение указателя
- Реализация конкретных потоков в его наследниках
- Классы обертки, облегчающие работу с потоками. …**Reader** / ….**Writer**
  - Чтение из / запись в поток
  - **BinaryReader / BinaryWriter** – бинарные чтение/запись
  - **StreamReader / StreamWriter** – текстовые чтение/запись
  - Понятие кодировки (**Encoding**)

Класс Stream

- Абстрактный класс. Реализации в потомках
- Чтение
  - int **Read**(byte[] buffer, int offset, int count)
  - int **ReadByte**()
- Запись
  - void **Write**(byte[] buffer, int offset, int count)
  - void **WriteByte**(byte value)
- Возможности
  - **CanRead** – можно ли читать
  - **CanWrite** – можно ли писать
  - **CanSeek** – можно ли двигать курсор
- Текущая позиция
  - long **Position** {get; set; }
- Перемещение
  - long **Seek**(long offset, SeekOrigin origin)
  - ;
- Сброс данных буферов чтения/записи
  - void **Flush**()
- Закрытие потока
  - void **Close**()
  - void **Dispose**()

Наследники Stream



Классы обертки

- Наследники **Stream**
  - **CryptoStream** – предоставляет шифрование потока
  - **BufferedStream** – добавляет буферизацию
  - **GZipStream**, **DeflateStream** – предоставляют возможность сжатия потока
- Предоставляют удобные способы чтения/записи в поток
  - **BinaryReader** / **BinaryWriter** – в бинарном виде
  - **StreamReader** / **StreamWriter** – в текстовом формате

FileStream

- Открытие файла
  - FileStream fileStream = **new FileStream**(@"d:\test.txt", FileMode.Create, FileAccess.Write);
    - FileMode: CreateNew, Create, Open, OpenOrCreate, Truncate, Append
    - FileAccess: Read, Write, ReadWrite
  - FileStream fileStream = **File.Create**(@"d:\test.txt");
  - **File.Open**(…), **File.OpenRead**(…), **File.OpenWrite**(…)
  - **fileInfo.Open**(…), **fileInfo.OpenRead**(…), **fileInfo.OpenWrite**(…), **fileInfo.Create**(…)

- Чтение, запись – стандартные методы Stream
  - **Read**() / **Write**() – чтение запись массива байт
  - **ReadByte**() / **WriteByte**() - чтение запись одного байта
  - Также есть набор асинхронных методов чтения/записи

- Пример (без обработки исключений)

```
FileStream fileStream = new FileStream(@"d:\test.txt", FileMode.Create);
string s = "Hello";
byte[] data = Encoding.UTF8.GetBytes(s);
fileStream.Write(data, 0, data.Length);
fileStream.Close();
```

StreamReader / StreamWriter

- Классы обертки. Оборачивают произвольный поток
- Предоставляют текстовый доступ к потоку
- Чтение/запись строк
  - StreamReader
    - **ReadLine**() – чтение строки
    - **ReadToEnd**() – сразу весь файл
  - StreamWriter
    - **Write**() – запись строки
    - **WriteLine**() – запись строки + символ новой строки
  - Параметры аналогичны функциям класса **Console**
- Close(), Dispose() – закрывают нижележащий поток

- Пример (без обработки исключение)

```
FileStream fileStream = new FileStream(@"d:\test.txt", FileMode.Open);
StreamReader sr = new StreamReader(fileStream);
while (!sr.EndOfStream) Console.WriteLine(sr.ReadLine());
sr.Close(); // Закроет и fileStream
```

- Пример

```
using (StreamReader sr = new StreamReader(fullFileName)) // за кулисами создастся FileStream
{
    while (!sr.EndOfStream) Console.WriteLine(sr.ReadLine());
} // sr.Close() будет автоматически вызван из метода Dispose при выходе из блока using { }
```

BinaryReader / BinaryWriter

- Классы обертки. Оборачивают произвольный поток
- Предоставляют бинарный доступ к потоку
- Чтение/ запись встроенных типов
  - ReadXXX() / WriteXXX()
  - XXX = Int32, Single, Double, …
  - ReadInt32(), WriteDouble() ….
- «Подсмотреть» следующий символ
  - PeekChar()
- Кодировка по умолчанию UTF8
  - Convert.ToBase64String(), Convert.FromBase64String и другие

Кодировка

- Класс **System.Text.Encoding**
- Представляет кодировку символов/кодировщик

- Получение кодировщика:
  - Статические свойства класса **Encoding**
    - Encoding.UTF7
    - **Encoding.UTF8** (предпочтительная кодировка)
    - Encoding.UTF32
    - Encoding.Unicode – представляет кодировку UTF16 (стандартная для строк .NET)
    - Encoding.ASCII

  - **Encoding GetEncoding()** получает кодировку по
    - Имени: GetEncoding(“windows-1251”)
    - Идентификатору кодовой страницы: GetEncoding(1251)

- Кодирование:
  - byte[] data = Encoding.UTF8.**GetBytes**(“Hello”);

- Декодирование:
  - string s = Encoding.UTF8.**GetString**(data);

- Использование кодировки:
  - StreamReader sr = new StreamReader(stream, Encoding.Unicode);
  - По умолчанию используется **UTF8**

- Для кодировок **Base64** используйте другой класс - **Convert**
  - Convert.ToBase64String(),
  - Convert.FromBase64String и другие

Текстовый ввод-вывод

- **TextReader/TextWriter** – базовые классы
  - **StreamReader/StreamWriter** – из потока
  - **StringReader/StringWriter** – из строки

- Стандартные потоки
  - **Console.In** – поток ввода
  - **Console.Out** – поток вывода
  - **Console.Error** – поток ошибок

Демонстрации

Текстовый ввод-вывод

Шифрование

- Класс **CryptoStream** – поток для криптографических преобразований
- Создание
  - Конструктор CryptoStream(Stream, ICryptoTransform, CryptoStreamMode)
    - Stream – исходный поток
    - ICryptoTransform – преобразователь (Encryptor/Decryptor)
    - CryptoStreamMode – Read / Write

- Пример

```
using (Aes aes = Aes.Create())
{
    aes.Key = Key;    aes.IV = Iv; // инициализация AES алгоритма
    using (FileStream inStream = new FileStream(@"d:\Encrypted.txt“, FileMode.Create)) // in поток
    {
        using (CryptoStream cryptoStream =   new CryptoStream(inStream,  aes.CreateEncryptor(), CryptoStreamMode.Write))
          // Шифрование. Обертка out потока
        {
                using (StreamWriter sw = new StreamWriter(cryptoStream))
                {
                    sw.WriteLine("Hello");                    // Запись в поток данных
                    sw.WriteLine("Hello again");              // Запись в поток данных
    }}}}
```

Криптография

- Реализации семеричных алгоритмов шифрования
  - Классы наследники от **Aes**, DES, TripleDES, RC2, Rijndael
- Реализация алгоритмов хеширования - наследники от класса HashAlgorithm
  - Хеширование: классы наследники от MD5, SHA1, SHA256, SHA384, SHA512, RIPEMD160 (реализация алгоритма MD160), SHA-3-256, SHA-3-384, SHA-3-512 (.NET 8+)
  - HMAC: классы наследники от KeyedHashAlgorithm: HMACMD5, HMACSHA256, HMACSHA384, HMACSHA512, HMACRIPEMD160, HMACSHA3_384, HMACSHA3_512 (.NET 8+), Kmac128 , Kmac256 (.NET 9+), SHAKE128, SHAKE256 (.NET 8+)
- Реализация асимметричных алгоритмов
  - Наследники от ECDiffieHellman (Elliptic Curve Diffie-Hellman (ECDH)), ECDsa (Elliptic Curve Digital Signature Algorithm (ECDSA)), RSA, DSA
- Не забывать звать Clear() или Dispose() по завершении работы
- Пространство имен System.Security.Cryptography

Демонстрации

Шифрование AES

ZipFile

- Класс ZipFile - предоставляет статические методы для создания, извлечения и открытия ZIP-архивов.

- Создание ZIP-архива из каталога
  - ZipFile.CreateFromDirectory()
- Извлечение содержимого ZIP-архива в каталог
  - ZipFile.ExtractToDirectory()

- Открытие архива:
  - ZipArchive ZipFile.Open()
  - ZipArchive ZipFile.OpenRead()
- Добавление новых файлов в существующий ZIP-архив
  - ZipArchive.CreateEntry
- Получение файла в ZIP-архиве
  - ZipArchive.GetEntry
- Получение всех файлов в ZIP-архиве
  - ZipArchive.Entries
- Открытие потока для отдельного файла, содержащегося в ZIP-архиве
  - ZipArchiveEntry.Open
- Удаление файла из ZIP-архива
  - ZipArchiveEntry.Delete

ZipFile

```
string zipFileName = "test.zip";
string directory=Directory.GetParent(Environment.CurrentDirectory).Parent.FullName;
if (File.Exists(zipFileName)) File.Delete(zipFileName);

using (ZipArchive archive = ZipFile.Open(zipFileName, ZipArchiveMode.Create))
{
        foreach (string file in Directory.GetFiles(directory, "*.cs"))
        {
            archive.CreateEntryFromFile(file, Path.GetFileName(file));
        }
}

string tempDir = Path.Combine(Path.GetTempPath(), "ZippingFolder");
Directory.CreateDirectory(tempDir);

using (ZipArchive archive = ZipFile.OpenRead(zipFileName))
{
    foreach (ZipArchiveEntry entry in archive.Entries)
    {
        if (entry.FullName.StartsWith("Prog", StringComparison.OrdinalIgnoreCase))
        {
            string destinationPath = Path.GetFullPath(Path.Combine(tempDir, entry.FullName));
            entry.ExtractToFile(destinationPath);
        }
    }
}
```

ZipArchve – представляет файл Zip архива

ZipArchiveEntry – представляет отдельный сжатый файл внутри ZIP-архива

Демонстрации

Zip