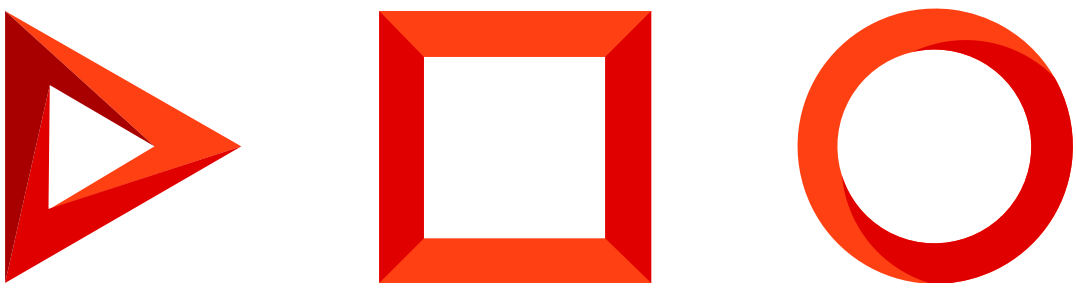
Creatio Academy

Back-end разработка

API для работы с файлами

Версия 8.0



Эта документация предоставляется с ограничениями на использование и защищена законами об интеллектуальной собственности. За исключением случаев, прямо разрешенных в вашем лицензионном соглашении или разрешенных законом, вы не можете использовать, копировать, воспроизводить, переводить, транслировать, изменять, лицензировать, передавать, распространять, демонстрировать, выполнять, публиковать или отображать любую часть в любой форме или посредством любые значения. Обратный инжиниринг, дизассемблирование или декомпиляция этой документации, если это не требуется по закону для взаимодействия, запрещены.

Информация, содержащаяся в данном документе, может быть изменена без предварительного уведомления и не может гарантировать отсутствие ошибок. Если вы обнаружите какие-либо ошибки, сообщите нам о них в письменной форме.

Содержание

API для работы с файлами

Средний

Начиная с версии 7.17.2 ядро Creatio предоставляет набор классов и интерфейсов по работе с файлами:

- пространство имен `Terrasoft.File.Abstractions` — интерфейсы и абстрактные классы, описывающие логику работы с файлами в Creatio.
- пространство имен `Terrasoft.File` — конкретные реализации абстракций, использующиеся в системе.

Местоположение файла и файловые локаторы

Местоположение файла в файловом хранилище задается с помощью **файлового локатора** — объекта, который реализует интерфейс `Terrasoft.File.Abstractions.IFileLocator`.

Файловый локатор обязательно содержит **уникальный идентификатор файла** `RecordId`.

Можно создавать разные реализации файловых локаторов для различных файловых хранилищ. В зависимости от специфики хранилища файловый локатор может содержать дополнительные свойства, позволяющие определить место хранения файла. Например, класс `Terrasoft.File.EntityFileLocator` — это реализация файлового локатора для текущего файлового хранилища Creatio [*Файлы и ссылки*] ([*Attachments*]). Объект класса `EntityFileLocator`, кроме свойства `RecordId`, имеет свойство `EntitySchemaName` — имя схемы объекта, в котором хранится файл, например: "ActivityFile" или "CaseFile".

Все методы по работе с файлами оперируют с файловыми локаторами.

Файлы и файловые хранилища

Структура файла в Creatio:

- метаданные файла;
- контент файла.

Метаданные описывают свойства файла:

- имя;
- размер в байтах;
- дата создания и т. п.

Основой для метаданных файла является абстрактный класс `Terrasoft.File.Abstractions.Metadata.FileMetadata`. Примером его конкретной реализации является класс `Terrasoft.File.Metadata.EntityFileMetadata` для описания метаданных файлов в объекте [*Файлы и ссылки*] ([*Attachments*]).

Контент — это непосредственно содержимое файла.

Метаданные файла и его контент хранятся в Creatio в разных **хранилищах**:

- Хранилище метаданных файлов должно реализовывать интерфейс `Terrasoft.File.Abstractions.Metadata.IFileMetadataStorage`.
- Хранилище контента файла должно реализовывать интерфейс `Terrasoft.File.Abstractions.Metadata.IFileContentStorage`.

Конкретные реализации этих интерфейсов скрывают в себе нюансы взаимодействия с различными системами хранения файлов: [*Файлы и ссылки*] ([*Attachments*]) Creatio, файловая система сервера, Amazon S3, Google Drive и т. п.

Интерфейс `Terrasoft.File.Abstractions.IFile` предоставляет необходимые методы для работы с файлами, хранящимися в любых типах файловых хранилищ. Реализация этого интерфейса означает **"файл"** в терминах Creatio. Методы в этом интерфейсе обеспечивают асинхронную работу с файлами. Синхронные версии этих методов находятся в расширяющем классе `Terrasoft.File.Abstractions.FileUtils`.

Получить файл (`IFileFactory`)

Интерфейс `Terrasoft.File.Abstractions.IFileFactory` предоставляет методы для создания и получения объектов некоторого класса, реализующего интерфейс `Terrasoft.File.Abstractions.IFile`. Этот интерфейс реализует фабрика, доступ к которой обеспечивается через методы класса `Terrasoft.File.FileFactoryUtils`, расширяющего класс `UserConnection`. Соответственно, для успешной работы с файлами необходимо иметь экземпляр `UserConnection` или `SystemUserConnection`.

Место хранения нового или существующего файла однозначно определяется его файловым локатором. Для получения доступа к существующему файлу необходимо знать его уникальный идентификатор `RecordId`. Для нового файла этот идентификатор формируется самостоятельно и передается в метод по созданию локатора.

Реализовать и зарегистрировать новый тип файлового хранилища

Для реализации нового типа файлового хранилища необходимо:

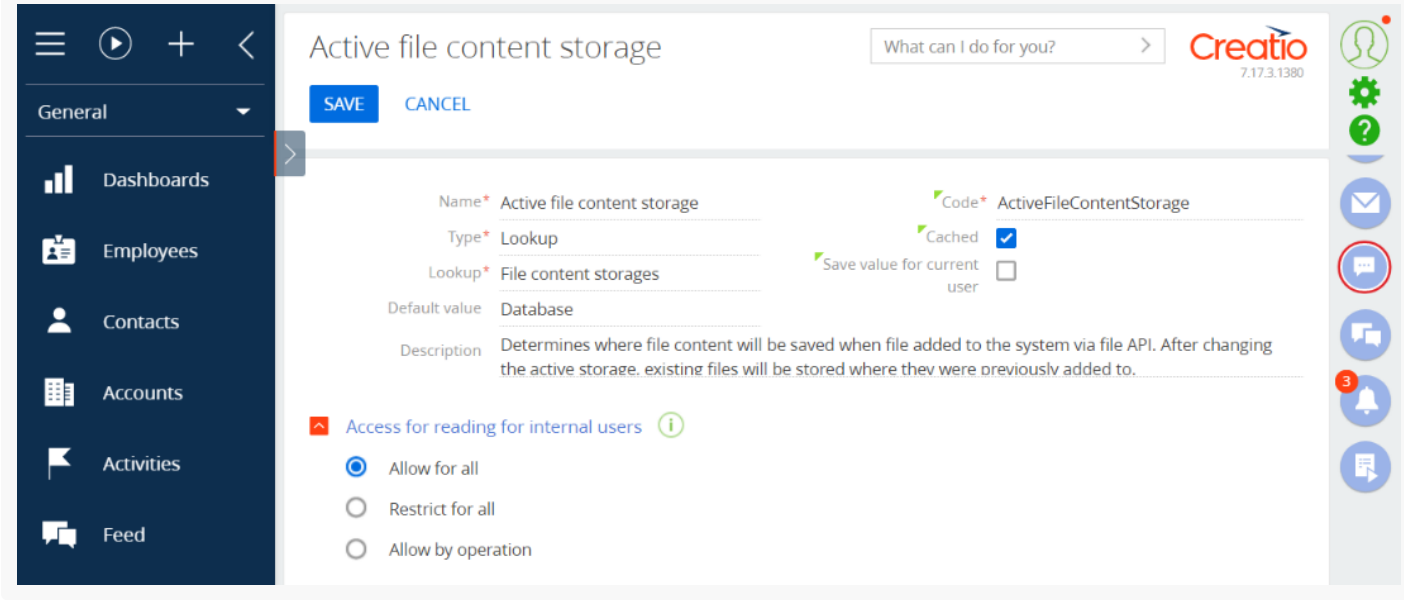
1. Создать свою реализацию интерфейса `Terrasoft.File.Abstractions.Content.IFileContentStorage`, который описывает необходимое API для работы с хранилищем файлового контента.
2. Если текущее хранилище метаданных файлов (`Terrasoft.File.Metadata.EntityFileMetadataStorage`) по каким-то причинам не подходит, необходимо реализовать собственное хранилище метаданных, свой тип метаданных и свой тип файлового локатора:
   - Хранилище данных должно реализовывать интерфейс `Terrasoft.File.Abstractions.Metadata.IFileMetadataStorage`.
   - Файловый локатор должен реализовывать интерфейс `Terrasoft.File.Abstractions.IFileLocator`.
   - Класс метаданных должен быть наследником абстрактного класса `Terrasoft.File.Abstractions.Metadata.FileMetadata`.
3. Новые хранилища контента и метаданных файлов необходимо зарегистрировать в соответствующих справочниках "SysFileContentStorage" и "SysFileMetadataStorage".

Исключения при работе с файлами

Тип исключения	Сообщение	Условия возникновения
`Terrasoft.File.Abstractions.FileNotFoundByLocatorException`	File not found by locator '{тип_локатора} {локатор.ToString}'	При доступе к любому из свойств или методов интерфейса `IFile`, если метаданные файла не найдены.
`System.InvalidOperationException`	Can't delete new file: '{тип_локатора}, {локатор.ToString}'	При попытке удаления только что созданного файла: `FileMetadata.StoringState == FileStoringState.New`
`Terrasoft.Common.NullOrEmptyException`	File name cannot be null or empty	При попытке сохранения файла с пустым полем `Name`
`System.InvalidOperationException`	Can't find a metadata storage for the '{тип_локатора}' locator type	Если подходящее по типу локатора хранилище метаданных файла не найдено
`System.InvalidOperationException`	Can't find a content storage for the '{тип_метаданных}' metadata type	Если подходящее по типу метаданных хранилище контента файла не найдено.

Настройка активного хранилища

Настройка активного хранилища происходит путем установки значения системной настройки [*Активное хранилище содержимого файлов*] (код `"ActiveFileContentStorage"`).



Примеры работы с файлами

Средний

Важно. Для корректной работы приведенных ниже примеров кода необходимо подключить пространство имен Terrasoft.Common.

```
using Terrasoft.Common;
```

Пример. Получить экземпляр класса, реализующего интерфейс IFile.

Получение IFile для существующего файла (вариант 1)

```
// Получаем фабрику файлов.
IFileFactory fileFactory = UserConnection.GetFileFactory();
// Создаем файловый локатор для файла с идентификатором recordId, который хранится в таблице БД
// "ActivityFile" ("Файлы и ссылки" для объекта "Activity").
var fileLocator = new EntityFileLocator("ActivityFile", recordId);
// Передаем сформированный локатор в метод Get фабрики.
IFile file = fileFactory.Get(fileLocator);
// В результате в file получаем экземпляр некоего класса, который реализует интерфейс IFile, чер
// можно производить другие манипуляции с файлом и его содержимым.
```

Получение IFile для существующего файла (вариант 2)

```
// Создаем файловый локатор для файла с идентификатором recordId, который хранится в таблице БД
// "ActivityFile" ("Файлы и ссылки" для объекта "Activity").
var fileLocator = new EntityFileLocator("ActivityFile", recordId);
// Передаем сформированный локатор в расширяющий метод GetFile.
IFile file = UserConnection.GetFile(fileLocator);
// В результате в file получаем экземпляр некоего класса, который реализует интерфейс IFile, чер
// можно производить другие манипуляции с файлом и его содержимым.
```

Получение IFile для нового файла (вариант 1)

```
// Получаем фабрику файлов.
IFileFactory fileFactory = UserConnection.GetFileFactory();
// Создаем уникальный идентификатор нового файла.
Guid recordId = Guid.NewGuid();
// Создаем файловый локатор для нового файла с идентификатором recordId, который хранится в табл
// "ActivityFile" ("Файлы и ссылки" для объекта "Activity").
var fileLocator= new EntityFileLocator("ActivityFile", recordId);
// Передаем сформированный локатор в метод Create фабрики.
IFile file = fileFactory.Create(fileLocator);
// В результате в file получаем экземпляр некоего класса, который реализует интерфейс IFile, чер
// можно производить другие манипуляции с файлом и его содержимым.
```

Получение IFile для нового файла (вариант 2)

```
// Создаем уникальный идентификатор нового файла.
Guid recordId = Guid.NewGuid();
// Создаем файловый локатор для нового файла с идентификатором recordId, который хранится в табл
// "ActivityFile" ("Файлы и ссылки" для объекта "Activity").
var fileLocator= new EntityFileLocator("ActivityFile", recordId);
// Передаем сформированный локатор в расширяющий метод CreateFile.
IFile file = UserConnection.CreateFile(fileLocator);
// В результате в file получаем экземпляр некоего класса, который реализует интерфейс IFile, чер
// можно производить другие манипуляции с файлом и его содержимым.
```

Пример. Создать новый файл с привязкой к записи раздела [*Активности*] ([*Activities*]).

Создание нового файла

```
// Создаем уникальный идентификатор нового файла.
Guid fileId = Guid.NewGuid();
// Создаем файловый локатор для нового файла.
var fileLocator= new EntityFileLocator("ActivityFile", fileId);
// Получаем объект IFile для нового файла.
IFile file = UserConnection.CreateFile(fileLocator);
// В хранилищах еще не сохранены ни метаданные нового файла, ни его контент.
// Задаем имя файлу в его метаданных.
file.Name = "New file";
// Устанавливаем новому файлу атрибут: привязываем этот файл к записи Активности с ключом activi
// Это тоже метаданные файла.
file.SetAttribute("ActivityId", activityId);
// Сохраняем метаданные файла. Это нужно делать обязательно ПЕРЕД сохранением его контента.
file.Save();
// byte[] content — это контент файла.
var content = new byte[] {0x12, 0x34, 0x56};
using (var stream = new MemoryStream(content)) {
    // Сохраняем контент в БД.
    // FileWriteOptions.SinglePart означает, что весь контент передается одним непрерывным кусо
    file.Write(stream, FileWriteOptions.SinglePart);
}
```

Пример. Получить содержимое файла.

Чтение содержимого файла

```
var content = new byte[]();
// Получаем файл по его локатору.
var fileLocator= new EntityFileLocator("ActivityFile", recordId);
IFile file = UserConnection.GetFile(fileLocator);
// Читаем все содержимое файла в массив байт content. Не забудьте освобождать объект потока с по
using (Stream stream = file.Read()) {
    content = stream.ReadAllBytes();
}
```

Пример. Скопировать файл, затем переместить его на новое место и удалить.

Копирование, перемещение, удаление файла

```
var content = new byte[]();
// Получаем файл по его локатору.
```

```
var fileLocator= new EntityFileLocator("ActivityFile", fileId);
IFile file = UserConnection.GetFile(fileLocator);
// Читаем все содержимое файла в массив байт content. Не забудьте освобождать объект потока с по
using (Stream stream = file.Read()) {
   content = stream.ReadAllBytes();
}

// Копирование файла.

// Создаем новый IFile для копирования текущего.
Guid copyFileId = Guid.NewGuid();
var copyFileLocator = new EntityFileLocator("ActivityFile", copyFileId);
IFile copyFile = UserConnection.CreateFile(copyFileLocator);
copyFile.Name = file.Name + " - Copy";
copyFile.Save();

// Копируем содержимого первого файла в новый файл.
copyFile.Write(new MemoryStream(content), FileWriteOptions.SinglePart);

// Альтернативный способ копирования файла.
file.Copy(copyFile);

// Перемещение файла.

// Создаем новый файл для перемещения текущего.
Guid moveFileId = Guid.NewGuid();
var moveFileLocator = new EntityFileLocator("ContactFile", moveFileId);
IFile moveFile = UserConnection.CreateFile(moveFileLocator);
moveFile.Save();

// Перемещаем на новое место.
file.Move(moveFile);

// Удаление исходного файла.
file.Delete();
```

Пример реализации файлового хранилища контента

Средний

Пример. Создать класс, реализующий интерфейс IFileContentStorage для создания хранилища контента в файловой системе.

Пример реализации файлового хранилища контента

```
namespace Terrasoft.Configuration
{
   using System.IO;
   using System.Threading.Tasks;
   using Terrasoft.File.Abstractions;
   using Terrasoft.File.Abstractions.Content;
   using Terrasoft.File.Abstractions.Metadata;
   using Terrasoft.File.Metadata;

   /// <summary>
   /// Хранилище контента в файловой системе.
   /// </summary>
   public class FsFileBlobStorage : IFileContentStorage
   {

      // Корневой путь к хранилищу.
      private const string BaseFsPath = "C:\\FsStore\\";

      private static string GetPath(FileMetadata fileMetadata) {
         var md = (EntityFileMetadata)fileMetadata;
         string key = $"{md.EntitySchemaName}\\{md.RecordId}_{fileMetadata.Name}";
         return Path.Combine(BaseFsPath, key);
      }

      public Task<Stream> ReadAsync(IFileContentReadContext context) {
         string filePath = GetPath(context.FileMetadata);
         Stream stream = System.IO.File.OpenRead(filePath);
         return Task.FromResult(stream);
      }

      public async Task WriteAsync(IFileContentWriteContext context) {
         string filePath = GetPath(context.FileMetadata);
         FileMode flags = context.WriteOptions != FileWriteOptions.SinglePart
            ? FileMode.Append
            : FileMode.OpenOrCreate;
         string dirPath = Path.GetDirectoryName(filePath);
         if (!Directory.Exists(dirPath)) {
            Directory.CreateDirectory(dirPath);
         }
         using (var fileStream = System.IO.File.Open(filePath, flags)) {
            context.Stream.CopyToAsync(fileStream);
         }
      }

      public Task DeleteAsync(IFileContentDeleteContext context) {
         string filePath = GetPath(context.FileMetadata);
```

```
            System.IO.File.Delete(filePath);
            return Task.CompletedTask;
        }

        public Task CopyAsync(IFileContentCopyMoveContext context) {
            string sourceFilePath = GetPath(context.SourceMetadata);
            string targetFilePath = GetPath(context.TargetMetadata);
            System.IO.File.Copy(sourceFilePath, targetFilePath);
            return Task.CompletedTask;
        }

        public Task MoveAsync(IFileContentCopyMoveContext context) {
            string sourceFilePath = GetPath(context.SourceMetadata);
            string targetFilePath = GetPath(context.TargetMetadata);
            System.IO.File.Move(sourceFilePath, targetFilePath);
            return Task.CompletedTask;
        }
    }
}
```

Интерфейс IFile C#

Средний

Пространство имен `Terrasoft.File.Abstractions`.

Интерфейс `Terrasoft.File.Abstractions.IFile` предоставляет необходимые методы для работы с файлами, хранящимися в любых типах файловых хранилищ. Методы этого интерфейса обеспечивают асинхронную работу с файлами. Синхронные версии этих методов находятся в расширяющем классе `Terrasoft.File.Abstractions.FileUtils`.

На заметку. Полное описание интерфейса `IFile` можно найти в документации ".NET библиотеки классов ядра платформы".

Свойства

`FileLocator` `IFileLocator`

Файловый локатор, который связан с текущим экземпляром класса, реализующего интерфейс `IFile`.

`Name` `string`

Имя файла.

`Length` `long`

Размер текущего файла в байтах.

`CreatedOn` `DateTime`

Дата и время создания текущего файла.

`ModifiedOn` `DateTime`

Дата и время модификации текущего файла.

`Exists` `bool`

Показывает существует ли текущий файл.

Методы

`Task CopyAsync(IFile target)`

Ассинхронно копирует текущий файл в новый `target`.

`Task MoveAsync(IFile target)`

Асинхронно перемещает текущий файл в новый `target`.

`Task DeleteAsync()`

Асинхронно удаляет текущий файл.

`Task WriteAsync(Stream stream, FileWriteOptions writeOptions)`

Асинхронно записывает содержимое текущего файла в поток `stream`.

`Task<Stream> ReadAsync()`

Асинхронно считывает содержимое текущего файла.

`Task SaveAsync()`

Асинхронно сохраняет метаданные текущего файла.

`void SetAttribute<TValue>(string name, TValue value)`

Устанавливает значение `value` атрибута `name` для текущего файла.

`TValue GetAttribute<TValue>(string name, TValue defaultValue)`

Возвращает значение `defaultValue` атрибута `name` либо значение по умолчанию для текущего файла.

Интерфейс IFileContentStorage C#

Средний

Пространство имен `Terrasoft.File.Abstractions`.

Интерфейс `Terrasoft.File.Abstractions.IFileContentStorage` предоставляет необходимые методы для работы с хранилищем контента файла.

На заметку. Полное описание интерфейса `IFileContentStorage` можно найти в документации ".NET библиотеки классов ядра платформы".

Методы

`Task<Stream> ReadAsync(IFileContentReadContext context)`

Считывает файловый контент.

`Task WriteAsync(IFileContentWriteContext context)`

Записывает файловый контент.

`Task DeleteAsync(IFileContentDeleteContext context)`

Удаляет файловый контент.

`Task CopyAsync(IFileContentCopyMoveContext context)`

Копирует файловый контент.

`Task MoveAsync(IFileContentCopyMoveContext context)`

Перемещает файловый контент.

Интерфейс IFileFactory C#

Средний

Пространство имен `Terrasoft.File.Abstractions`.

Интерфейс `Terrasoft.File.Abstractions.IFileFactory` предоставляет набор методов для получения или создания экземпляра класса, реализующего интерфейс `Terrasoft.File.Abstractions.IFile`.

На заметку. Полное описание интерфейса `IFileFactory` можно найти в документации ".NET библиотеки классов ядра платформы".

Свойства

`UseRights bool`

Определяет должны ли учитываться права пользователя при создании файла или нет.

Методы

`IFile Get(IFileLocator fileLocator, FileOptions options)`

Возвращает экземпляр класса, реализующего интерфейс `IFile`, с заданными параметрами `options` из определенного файлового локатора `fileLocator`.

`IFile Create(IFileLocator fileLocator, FileOptions options)`

Создает новый экземпляр класса, реализующего интерфейс IFile, с заданными параметрами `options` для определенного файлового локатора `fileLocator`.

Класс EntityFileLocator C#

Средний

Пространство имен `Terrasoft.File`.

Класс предоставляет реализацию интерфейса `IFileLocator` для текущего файлового хранилища Creatio.

На заметку. Полный перечень методов и свойств класса `EntityFileLocator`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntityFileLocator()`

Создает новый экземпляр класса `EntityFileLocator` .

`EntityFileLocator(string entitySchemaName, Guid recordId)`

Создает новый экземпляр класса `EntityFileLocator` для заданного файла `recordId`, привязанного к конкретной схеме объекта `entitySchemaName`.

Свойства

`EntitySchemaName string`

Метаданные — имя схемы объекта, в котором хранится файл.

`RecordId Guid`

Метаданные — идентификатор файла.

Класс EntityFileMetadata C#

Средний

Пространство имен `Terrasoft.File`.

Класс `Terrasoft.File.EntityFileMetadata` реализует абстрактный класс Terrasoft.File.Abstractions.Metadata.FileMetadata. Этот класс описывает метаданные файлов в объекте [*Файлы и ссылки*] ([*Attachments*]) и предоставляет методы для работы с ними.

На заметку. Полный перечень методов и свойств класса `EntityFileMetadata`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntityFileMetadata(EntityFileLocator fileLocator)`

Создает новый экземпляр класса `EntityFileMetadata` для файлового локатора `fileLocator`.

Свойства

`Attributes` `IReadOnlyDictionary<string, object>`

Коллекция значений аттрибутов.

`RecordId` `Guid`

Идентификатор файла.

`EntitySchemaName` `string`

Имя схемы объекта, в котором хранится файл.

Методы

`override void SetAttribute<TValue>(string name, TValue value)`

Устанавливает для файла дополнительный значение `value` дополнительного аттрибута `name`.

`override TValue GetAttribute<TValue>(string name, TValue defaultValue)`

Возвращает установленное значение либо значение по умолчанию `defaultValue` определенного дополнительного аттрибута `name`.

Класс FileFactoryUtils C#

Средний

Пространство имен `Terrasoft.File`.

Класс `Terrasoft.File.FileFactoryUtils` предоставляет расширяющие методы класса `UserConnection` и класса-фабрики, реализующего интерфейс `Terrasoft.File.AbstractionsIFileFactory`. Таким образом класс обеспечивает доступ к фабрике создания новых или получения существующих файлов. Соответственно, для успешной работы с файлами необходимо иметь экземпляр `UserConnection` или `SystemUserConnection`.

На заметку. Полный перечень методов и свойств класса `FileFactoryUtils`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Методы

`static IFileFactory GetFileFactory(this UserConnection source)`

Расширяющий метод класса `UserConnection`, который возвращает экземпляр класса, реализующего интерфейс `IFileFactory`.

`static IFile GetFile(this UserConnection source, IFileLocator fileLocator)`

Расширяющий метод класса `UserConnection`, который возвращает экземпляр класса, реализующего интерфейс `IFile` из определенного файлового локатора `fileLocator`.

`static IFile CreateFile(this UserConnection source, IFileLocator fileLocator)`

Расширяющий метод класса `UserConnection`, который создает новый экземпляр класса, реализующего интерфейс `IFile` для определенного файлового локатора `fileLocator`.

`static IFile Get(this IFileFactory source, IFileLocator fileLocator)`

Расширяющий метод класса, реализующего интерфейс `IFileFactory`. Возвращает экземпляр класса, реализующего интерфейс `IFile` для определенного файлового локатора `fileLocator`.

`static IFile Create(this IFileFactory source, IFileLocator fileLocator)`

Расширяющий метод класса, реализующего интерфейс `IFileFactory`. Создает новый экземпляр класса, реализующего интерфейс `IFile` для определенного файлового локатора `fileLocator`.

`static IFileFactory WithRightsDisabled(this IFileFactory source)`

Расширяющий метод класса, реализующего интерфейс `IFileFactory`. Возвращает экземпляр класса, реализующего интерфейс `IFileFactory`, настроенного без учета прав пользователя.

Класс FileMetadata C#

Средний

Пространство имен `Terrasoft.File.Abstractions.Metadata`.

Абстрактный класс `Terrasoft.File.Abstractions.Metadata.FileMetadata` предоставляет свойства метаданных файла и методы для работы с метаданными.

На заметку. Полный перечень методов и свойств класса `FileMetadata`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Свойства

`Name` `string`

Имя файла.

`Length` `long`

Размер файла в байтах.

`CreatedOn` `DateTime`

Дата и время создания файла.

`ModifiedOn` `DateTime`

Дата и время изменения файла.

`FileContentStorageId` `Guid`

Идентификатор хранилища контента файла.

`StoringState` `FileStoringState`

Состояние файла ("Новый", "Изменен", "Неизменен", "Удален").

Методы

`abstract void SetAttribute<TValue>(string name, TValue value)`

Устанавливает для файла дополнительный значение `value` дополнительного аттрибута `name`.

`abstract TValue GetAttribute<TValue>(string name, TValue defaultValue)`

Возвращает установленное значение либо значение по умолчанию `defaultValue` определенного дополнительного аттрибута `name`.

`void SetStoringState(FileStoringState newState)`

Устанавливает состояние файла `FileStoringState.Modified` если предыдущее состояние не равно `FileStoringState.New`.

Класс FileUtils C#

Средний

Пространство имен `Terrasoft.File.Abstractions`.

Класс `Terrasoft.File.Abstractions.FileUtils` предоставляет расширяющие методы для работы с файлами.

На заметку. Полный перечень методов и свойств класса `FileUtils`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Методы

```
static void SetAttributes(this IFile source, IReadOnlyDictionary<string, object> attributes)
```

Устанавливает для файла значения аттрибутов, переданных в коллекции `attributes`.

```
static void Save(this IFile source)
```

Сохраняет метаданные файла.

```
static Stream Read(this IFile source)
```

Считывает содержимое файла.

```
static void Write(this IFile source, Stream stream, FileWriteOptions writeOptions)
static void Write(this IFile source, byte[] content)
```

Записывает содержимое файла.

Параметры

`source`	Файл, содержимое которого необходимо записать.
`stream`	Поток, предоставляющий содержимое файла.
`writeOptions`	Параметры для записи файла.
`content`	Содержимое файла в виде массива байтов.

```
static void Delete(this IFile source)
```

Удаляет определенный файл.

```
static void Copy(this IFile source, IFile target)
```

Копирует существующий файл `source` в новый `target`.

```
static void Move(this IFile source, IFile target)
```

Перемещает существующий файл `source` в новое место `target`.