Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное учреждение высшего профессионального образования

«Алтайский государственный технический университет им. И.И. Ползунова»

Факультет . информационных технологий

Кафедра . прикладная математика .

Курсовой проект защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_

Руководитель

проекта (работы) \_\_\_\_\_\_\_ Старолетов М.С.\_\_\_\_\_\_

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_2013 г.

**Реализация многопоточного сервера закачек**

**с возможностью удаленного управления**

Пояснительная записка

к курсовой работе

по дисциплине Проектирование сетевых и многопоточных приложений

**КР 230105.11.000ПЗ**

Студент группы \_\_\_\_\_\_\_ПОВТ-81\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Левина А.В.\_\_\_\_

Руководитель

проекта (работы) \_\_ \_\_\_\_\_\_\_\_\_\_ Старолетов С.М.\_\_

Барнаул 2013

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Задание на курсовую работу**  Учебная дисциплина: Проектирование сетевых и многопоточных приложений  ФИО студента: Левина Алёна Викторовна  Группа: ПОВТ-81  Задание: Реализация многопоточного сервера закачек с возможностью удаленного управления.  Руководитель: Старолетов С. М. | | | | | | | | | | |
|  |  |  |  |  | **КР 230105.11.000ПЗ** | | | | | |
|  |  |  |  |  |
| Изм | Лист | № докум. | Подпись | Дата |
| Разработал | | Левина А.В. |  |  | Реализация многопоточного сервера закачек с возможностью удаленного управления | Лит. | | | Лист | Листов |
| Пров. | | Старолетов С.М. |  |  |  | У |  | 2 | 43 |
|  | |  |  |  | АлтГТУ, ФИТ  *гр. ПОВТ-81* | | | | |
| Н.контролер | |  |  |  |
| Утв. | | Кантор С.А. |  |  |

**Аннотация**

В данной работе представлены сведения о проектировании и реализации многопоточного сервера закачек с возможностью удаленного управления. Приложение позволяет управлять удаленно сервером, то есть добавлять загрузку на скачивание, останавливать, ставить на паузу и возобновлять загрузку на сервере.

Программное обеспечение написано на языке программирования С# с использованием Microsoft Visual Studio 2010 и NET Framework 4. Исходный код приложения, а также данный отчет представлены на прилагаемом CD диске.

**Содержание**

[Введение 5](#_Toc336174043)

[1 Постановка задачи 6](#_Toc336174044)

[1.1 Развернутая постановка задачи 6](#_Toc336174045)

[2 Проектирование программного обеспечения 7](#_Toc336174046)

[2.1 Диаграмма вариантов использования 7](#_Toc336174047)

[2.2 Диаграмма развертывания 8](#_Toc336174048)

[2.3 Временная диаграмма 8](#_Toc336174049)

[2.4 Диаграмма деятельности 8](#_Toc336174050)

[3 Описание программного обеспечения 10](#_Toc336174051)

[3.1 Общий подход к реализации 10](#_Toc336174052)

[3.2 Описание приложения 10](#_Toc336174053)

[Заключение 32](#_Toc336174094)

[Список использованных источников 33](#_Toc336174095)

[Приложение А](#_Toc336174096) [Руководство пользователя 34](#_Toc336174097)

[Приложение Б](#_Toc336174098) [Текст программы 36](#_Toc336174099)

**Введение**

Многопоточные сервера используются в том случае, когда к одним и тем же данным нужно обратиться определенному числу клиентов. Главное, чтобы многопоточный сервер обрабатывал все запросы к нему поступающие, потому что сферы использование технологии разнообразны и обработка запросов не должна задерживать деятельность клиентов.

Написание многопоточного сервера возможно разными способами с использованием сокетов.

Сокеты — это транспортный механизм, который чаще всего используется в высокопроизводительных серверных приложениях. Библиотека Win32 Windows Sockets (Winsock) предоставляет механизмы, которые повышают быстродействие программ, использующих сокеты, а в Microsoft .NET Framework имеется более высокий по отношению к Winsock уровень, благодаря чему управляемые приложения также могут взаимодействовать через сокеты.

.NET предоставляет более абстрактные классы вроде TcpListener и TcpClient (тоже относящиеся к пространству имен System.Net.Sockets), они не поддерживают некоторые возможности, предлагаемые более низкоуровневым классом Socket. Тем не менее, они полезны во многих ситуациях. У класса TcpListener есть простые методы, позволяющие прослушивать и принимать входящие запросы на соединение в блокирующем синхронном режиме, а у класса TcpClient — простые методы установления соединения, передачи и приема потоковых данных по сети в синхронном блокирующем режиме. Поэтому для данного приложения использовался этот класс.

В таком приложении на сервер устанавливается серверная часть приложения, а на компьютеры клиентов клиентская часть. Клиент удаленно, посредством программы, скачивает определенные файлы, которые будут храниться на сервере.

# 1 Постановка задачи

## 1.1 Развернутая постановка задачи

Реализация многопоточного сервера закачек с возможностью удаленного управления.

Сервер закачек получает от клиента команды по закачиванию файла с некоторого заданного адреса в Интернете, добавляет в очередь и скачивает их в несколько потоков (также, несколько файлов одновременно). Число одновременно закачиваемых файлы и количество потоков на файл задается. Как файл скачан, клиент оповещается об этом.

# 2 Проектирование программного обеспечения

## 2.1 Диаграмма вариантов использования

На рисунке 2.1 представлена диаграмма использования для нашего приложения.

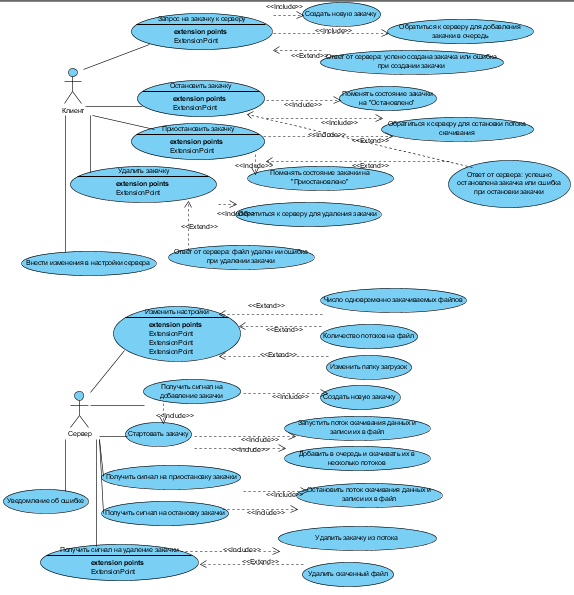


Рисунок 2.1 - Диаграмма вариантов использования

## 2.2 Диаграмма развертывания

На рисунке 2.2 представлена диаграмма развертывания приложения. В ней отраженны необходимые файлы и библиотеки для функционирования программы.

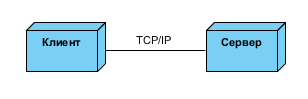


Рисунок 2.2 - Диаграмма развертывания

## 2.3 Временная диаграмма

На рисунке 2.3 представлена временная диаграмма. Данная диаграмма отображает наиболее полный вариант работы программы.

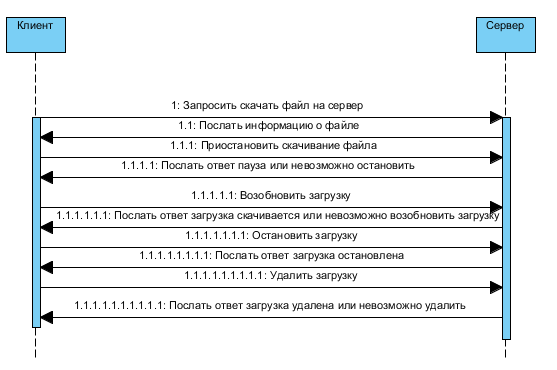


Рисунок 2.3 - Временная диаграмма

## 2.4 Диаграмма деятельности

На рисунке 2.4 представлена диаграмма деятельности.

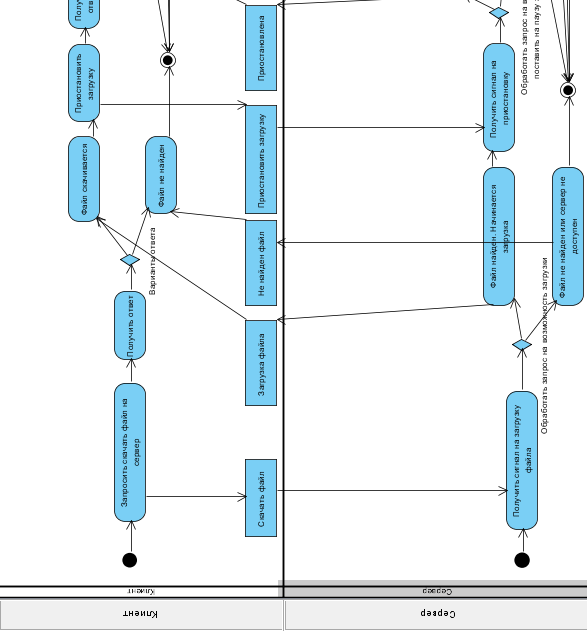
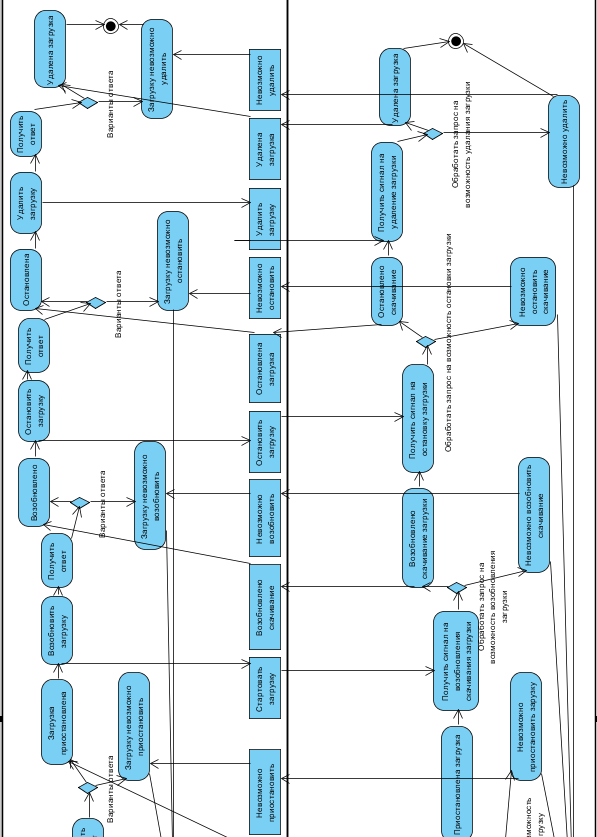


Рисунок 2.4 - Диаграмма деятельности

# 3 Описание программного обеспечения

## 3.1 Общий подход к реализации

Данный программный продукт представляет собой клиент-серверное приложение, написанное с использованием библиотеки .NET Framework 4 на языке C#. Приложение были написано в среде Microsoft Visual Studio 2010.

Приложение было разделено на три части:

* серверная часть;
* клиентская часть;
* библиотека совместно используемых классов.

Серверная часть приложения отвечает за управлениями загрузками (скачивание, остановка загрузки, пауза и возобновление скачивания, удаление загрузки). Сервер является многопоточным: взаимодействие с каждым клиентом происходит в отдельном потоке. Подключения принимаются также в отдельном потоке, а за отправку команд клиентам отвечает главный поток приложения.

Клиентская часть приложения отвечает за отправку сигналов серверу о том, что требуется скачать, а также с этой загрузкой возможно работать. Все, что происходит с загрузкой, отображается в клиентском приложении.

C# — язык программирования, сочетающий объектно-ориентированные и контекстно-ориентированные концепции. Язык имеет строгую статическую типизацию, поддерживает полиморфизм, перегрузку операторов, указатели на функции-члены классов, атрибуты, события, свойства, исключения, комментарии в формате XML. На C# легко реализуется графический пользовательский интерфейс, поэтому он был выбран для реализации приложения.

## 3.2 Описание приложения

На рисунке 3.1 приведена диаграмма основных классов, реализующих взаимодействие между сервером и клиентами.

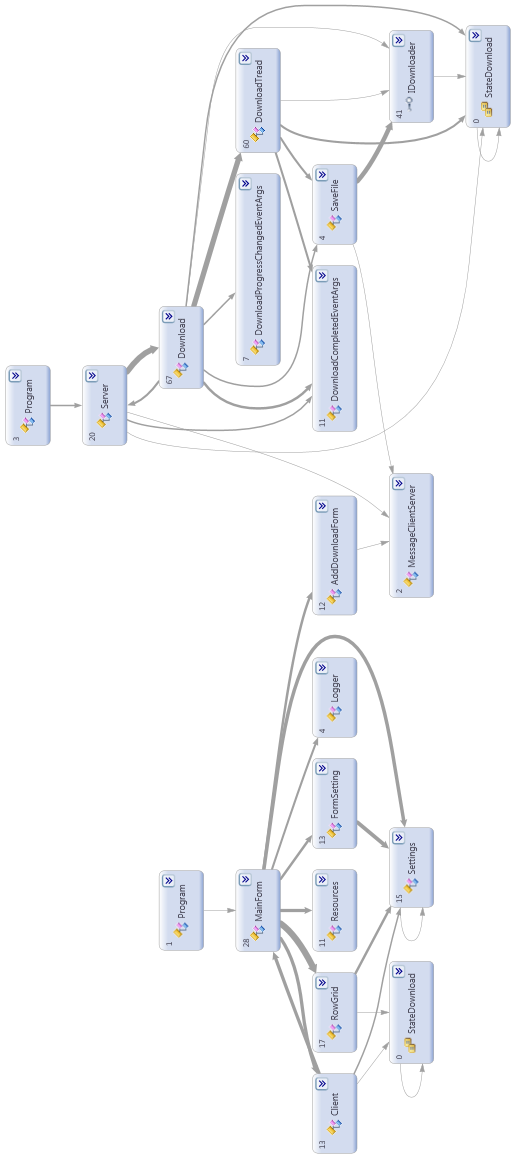


Рисунок 3.1 – Схема программы

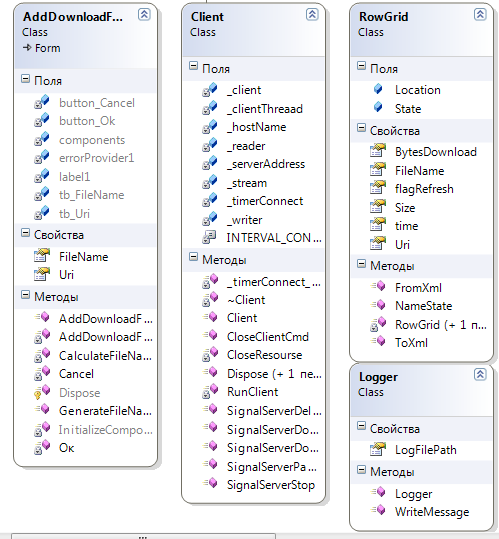
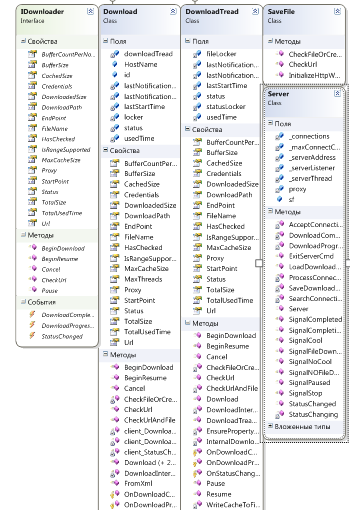


Рисунок 3.2 - Диаграмма основных классов

Документация сгенерирована с использованием программы Doxygen.

**Package Download**

|  |  |
| --- | --- |
| Packages | |
| package | [**Properties**](file:///C:\Users\Алёна\Desktop\html\namespace_download_1_1_properties.html) |
| Data Structures | |
| class | [**AddDownloadForm**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_add_download_form.html) |
| class | [**Client**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_client.html) |
| class | [**FormSetting**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_form_setting.html) |
| class | [**Logger**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_logger.html) |
| class | [**MainForm**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_main_form.html) |
| class | **Program** |
| class | [**RowGrid**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html) |
| Enumerations | |
| enum | **StateDownload** {    **Completed** = 0, **Downloading** = 1, **Paused** = 2, **Created** = 3,    **Stopped** = 4, **Nofile** = 5  } |

**Download.Client Class Reference**

|  |  |
| --- | --- |
| Public Member Functions | |
|  | [**Client**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_client.html#aa6e58e2f03686ad89d6391751430f198) (IPEndPoint serverAddress) |
|  | Клиент |
| void | [**SignalServerDownload**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_client.html#a372c449629657a9f120fa32dde2f3717) (string strUri) |
|  | Отправка на сервер сигнала на загрузку файла |
| void | [**SignalServerDownloadLast**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_client.html#a0241ff10ab556b3c66daf030516b80b9) (string strUri) |
|  | Отправка на сервер сигнала на загрузку имеющегося файла |
| void | [**SignalServerPause**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_client.html#ab494d25a513d2a0ecd9dae6b748f5374) (string strUri) |
|  | Отправка на сервер сигнала на установку на паузу загрузку файла |
| void | [**SignalServerStop**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_client.html#aa80291c0c6b86209256002e38fa9d3a4) (string strUri) |
|  | Отправка на сервер сигнала остановки загрузки файла |
| void | [**SignalServerDelete**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_client.html#a3979d0454478f252dddb80f7318f9143) (string strUri) |
|  | Отправка на сервер сигнала удаление загрузки файла |
| void | [**CloseClientCmd**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_client.html#a6950cbfce2d7a69b44a17e6d84603662) () |
|  | Отправка на сервер команды об отключении клиента |
| void | **Dispose** () |
| Protected Member Functions | |
| virtual void | **Dispose** (bool disposing) |

## Constructor & Destructor Documentation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [**Download.Client.Client**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_client.html#aa6e58e2f03686ad89d6391751430f198) | **(** | **IPEndPoint** | **serverAddress** | **)** |  |

Клиент

**Parameters:**

|  |  |
| --- | --- |
| **serverAddress** | Адрес сервера |

## Member Function Documentation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **void** [**Download.Client.CloseClientCmd**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_client.html#a6950cbfce2d7a69b44a17e6d84603662) | **(** |  | **)** |  |

Отправка на сервер команды об отключении клиента

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **void** [**Download.Client.SignalServerDelete**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_client.html#a3979d0454478f252dddb80f7318f9143) | **(** | **string** | **strUri** | **)** |  |

Отправка на сервер сигнала удаление загрузки файла

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **void** [**Download.Client.SignalServerDownload**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_client.html#a372c449629657a9f120fa32dde2f3717) | **(** | **string** | **strUri** | **)** |  |

Отправка на сервер сигнала на загрузку файла

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **void** [**Download.Client.SignalServerDownloadLast**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_client.html#a0241ff10ab556b3c66daf030516b80b9) | **(** | **string** | **strUri** | **)** |  |

Отправка на сервер сигнала на загрузку имеющегося файла

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **void** [**Download.Client.SignalServerPause**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_client.html#ab494d25a513d2a0ecd9dae6b748f5374) | **(** | **string** | **strUri** | **)** |  |

Отправка на сервер сигнала на установку на паузу загрузку файла

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **void** [**Download.Client.SignalServerStop**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_client.html#aa80291c0c6b86209256002e38fa9d3a4) | **(** | **string** | **strUri** | **)** |  |

Отправка на сервер сигнала остановки загрузки файла

**Download.RowGrid Class Reference**

|  |  |
| --- | --- |
| Public Member Functions | |
|  | **RowGrid** ([**Uri**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#ac0d646272b0729050841ef36a796321b) \_uri, string \_filename) |
| void | **ToXml** (XmlWriter writer) |
| string | **NameState** (StateDownload stateDownload) |
| Static Public Member Functions | |
| static [**RowGrid**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html) | **FromXml** (XmlReader reader) |
| Data Fields | |
| StateDownload | [**State**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#a58ec6791b0f5466d14456802f64ac950) |
|  | Состояние загрузки |
| string | [**Location**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#a3f3d5d4e4221e7d0616c000477f62055) = Properties.Settings.Default.DownloadServer |
|  | Раcположение(на какой сервер идет закачка) |
| Properties | |
| [**Uri**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#ac0d646272b0729050841ef36a796321b) | [**Uri**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#ac0d646272b0729050841ef36a796321b) [get, set] |
|  | Идентификатор ресурса |
| string | [**FileName**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#a5036a519f30946aa9afb1205ba0554a8) [get, set] |
|  | Имя скачиваемого файла |
| long | [**Size**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#ad62505ddee288a08858ffec38ee49779) [get, set] |
|  | Размер скачиваемого файла |
| long | [**BytesDownload**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#adf8fbe0c486869de4c66764d0d69ae54) [get, set] |
|  | Скачано байт |
| bool | [**flagRefresh**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#a2b1a7eaaec2c87aef957eb580c235063) [get, set] |
|  | Флаг обновления |
| TimeSpan | [**time**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#adce7ef42978f00838347856839c6581b) [get, set] |
|  | Время затрачиваемое на загрузку |

## Field Documentation

|  |
| --- |
| **string** [**Download.RowGrid.Location**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#a3f3d5d4e4221e7d0616c000477f62055) **= Properties.Settings.Default.DownloadServer** |

Раcположение(на какой сервер идет закачка)

|  |
| --- |
| **StateDownload** [**Download.RowGrid.State**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#a58ec6791b0f5466d14456802f64ac950) |

Состояние загрузки

## Property Documentation

|  |
| --- |
| **long** [**Download.RowGrid.BytesDownload**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#adf8fbe0c486869de4c66764d0d69ae54) **[get, set]** |

Скачано байт

|  |
| --- |
| **string** [**Download.RowGrid.FileName**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#a5036a519f30946aa9afb1205ba0554a8) **[get, set]** |

Имя скачиваемого файла

|  |
| --- |
| **bool** [**Download.RowGrid.flagRefresh**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#a2b1a7eaaec2c87aef957eb580c235063) **[get, set]** |

Флаг обновления

|  |
| --- |
| **long** [**Download.RowGrid.Size**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#ad62505ddee288a08858ffec38ee49779) **[get, set]** |

Размер скачиваемого файла

|  |
| --- |
| **TimeSpan** [**Download.RowGrid.time**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#adce7ef42978f00838347856839c6581b) **[get, set]** |

Время затрачиваемое на загрузку

|  |
| --- |
| [**Uri**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#ac0d646272b0729050841ef36a796321b)[**Download.RowGrid.Uri**](file:///C:\Users\Алёна\Desktop\html\class_download_1_1_row_grid.html#ac0d646272b0729050841ef36a796321b) **[get, set]** |

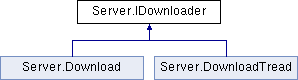
Идентификатор ресурса

**Package Server**

|  |  |
| --- | --- |
| Data Structures | |
| class | [**Download**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html) |
| class | [**DownloadTread**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html) |
| interface | [**IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html) |
| class | [**DownloadProgressChangedEventArgs**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_progress_changed_event_args.html) |
| class | [**DownloadCompletedEventArgs**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_completed_event_args.html) |
| class | [**Program**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_program.html) |
| class | [**SaveFile**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_save_file.html) |
| class | [**Server**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html) |

**Server.IDownloader Interface Reference**

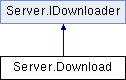
Inheritance diagram for Server.IDownloader:



|  |  |
| --- | --- |
| Public Member Functions | |
| void | **CheckUrl** (out string fileName) |
| void | **BeginDownload** () |
| void | **Pause** () |
| void | **BeginResume** () |
| void | **Cancel** () |
| Properties | |
| Uri | **Url** [get] |
| string | **DownloadPath** [get, set] |
| long | **TotalSize** [get, set] |
| ICredentials | **Credentials** [get, set] |
| IWebProxy | **Proxy** [get, set] |
| bool | **IsRangeSupported** [get, set] |
| long | **StartPoint** [get, set] |
| long | **EndPoint** [get, set] |
| string | **FileName** [get, set] |
| long | **DownloadedSize** [get] |
| int | **CachedSize** [get] |
| bool | **HasChecked** [get, set] |
| StateDownload | **Status** [get, set] |
| TimeSpan | **TotalUsedTime** [get] |
| int | **BufferSize** [get, set] |
| int | **BufferCountPerNotification** [get, set] |
| int | **MaxCacheSize** [get, set] |
| Events | |
| EventHandler < [**DownloadCompletedEventArgs**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_completed_event_args.html) > | **DownloadCompleted** |
| EventHandler < [**DownloadProgressChangedEventArgs**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_progress_changed_event_args.html) > | **DownloadProgressChanged** |
| EventHandler | **StatusChanged** |

**Server.Download Class Reference**

Inheritance diagram for Server.Download:



|  |  |
| --- | --- |
| Public Member Functions | |
|  | **Download** (string uri) |
|  | **Download** (string uri, int maxThreadCount) |
| void | [**CheckUrlAndFile**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a931f2baa0e76732cfb2364f7ba331a32) (out string fileName) |
|  | проверить Uri и файл |
| void | [**CheckUrl**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a0e697c2683f23b26eab39bbea6ecd5e4) (out string fileName) |
|  | Проверить uri. |
| void | [**BeginDownload**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a3bd4924724a1d9fcae8205fd39b6c693) () |
|  | Начать загрузку |
| void | [**Pause**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a6a0b632dc280814176bc62dfde273ea5) () |
|  | Пауза |
| void | [**BeginResume**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a93381d8ffa2478dd945af7bcfe8b8eac) () |
| void | [**Cancel**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#ae57f49c37094469d9ff8cb2235f8ee3a) () |
|  | Отмена загрузки |
| void | [**ToXml**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a2c228786173a0ce7a7103332c0571c3c) (XmlWriter writer) |
|  | Записывает данные о закачке из xml файла |
| Static Public Member Functions | |
| static [**Download**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html) | [**FromXml**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#ad75d67c74f8c87e8cf637d3197ea349f) (XmlReader reader) |
|  | Загужает данные о закачке из xml файла |
| Data Fields | |
| int | **id** |
| String | **HostName** |
| Protected Member Functions | |
| virtual void | [**OnStatusChanged**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a97782a8b48d900a97ba7ed4871da7496) (EventArgs e) |
|  | Событие StatusChanged. |
| virtual void | [**OnDownloadCompleted**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a5efcf9fd790ad2ae716ae028a10af62b) ([**DownloadCompletedEventArgs**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_completed_event_args.html) e) |
|  | Событие DownloadCompleted. |
| virtual void | [**OnDownloadProgressChanged**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a6d1d398e2c7ee312d31e4f72420b73a7) ([**DownloadProgressChangedEventArgs**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_progress_changed_event_args.html) e) |
|  | Событие DownloadProgressChanged event. |
| Properties | |
| string | **DownloadPath** [get, set] |
| string | [**FileName**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#acbc8c865e87fed492247843a18d26158) [get, set] |
|  | Имя скачиваемого файла |
| long | [**TotalSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a6408eb3ab98fbdc5893bb4bdeaebd422) [get, set] |
|  | Размер скачиваемого файла |
| Uri | [**Url**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#aa7b186737ef695e09cda2accc24d0cf1) [get, set] |
|  | Идентификатор ресурса |
| StateDownload | **Status** [get, set] |
| int | **MaxThreads** [get, set] |
| int | [**BufferCountPerNotification**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a8f76f908ea8c1fe0d7044a9af800fd97) [get, set] |
|  | Если получим число буферов, затем возникает событие DownloadProgressChanged. |
| int | [**BufferSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a114b86d2e3f4628baad9b92653584d9a) [get, set] |
|  | Установить BufferSize при чтении данных в поток ответа. |
| int | [**MaxCacheSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a90d155c9b484afd9d8af34aea00e45d9) [get, set] |
|  | Размер кэша в памяти. |
| long | [**StartPoint**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a36a2fde5cda1a65d98fb84877974b149) [get, set] |
|  | Начальная точка |
| long | [**EndPoint**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#ad4b365d2703c3d38db2dac2c00f016cf) [get, set] |
|  | Конечная точка |
| bool | [**HasChecked**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a4a832f971d14e7ba901734a59c83ac72) [get, set] |
|  | Флаг возможности загрузки |
| bool | [**IsRangeSupported**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#ab33683e17870b994a91b2e09b1678442) [get, set] |
|  | Укажите, следует ли удаленный сервер поддерживает "Accept-Ranges" заголовка. Используйте CheckUrl метод для инициализации этого свойства внутренне. |
| ICredentials | [**Credentials**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#affbbae50bbff6f235f7ebc1591eb70ff) [get, set] |
|  | Сетевые учетные данные, используемые для проверки подлинности запроса |
| int | **CachedSize** [get, set] |
| IWebProxy | [**Proxy**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#ae6a6197e06936067b3740dfdf2863688) [get, set] |
|  | Прокси |
| long | [**DownloadedSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a3506f35159b05d03c667e0577daff6c7) [get] |
|  | размер загруженной чати файла |
| TimeSpan | [**TotalUsedTime**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a0e61070551f00134e0109bda665287bb) [get] |
|  | Использованное время |
| Events | |
| EventHandler < [**DownloadProgressChangedEventArgs**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_progress_changed_event_args.html) > | **DownloadProgressChanged** |
| EventHandler < [**DownloadCompletedEventArgs**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_completed_event_args.html) > | **DownloadCompleted** |
| EventHandler | **StatusChanged** |

## Member Function Documentation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **void** [**Server.Download.BeginDownload**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a3bd4924724a1d9fcae8205fd39b6c693) | **(** |  | **)** |  |

Начать загрузку

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **void** [**Server.Download.BeginResume**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a93381d8ffa2478dd945af7bcfe8b8eac) | **(** |  | **)** |  |

Возобновить скачивание

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **void** [**Server.Download.Cancel**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#ae57f49c37094469d9ff8cb2235f8ee3a) | **(** |  | **)** |  |

Отмена загрузки

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **void** [**Server.Download.CheckUrl**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a0e697c2683f23b26eab39bbea6ecd5e4) | **(** | **out string** | **fileName** | **)** |  |

Проверить uri.

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **void** [**Server.Download.CheckUrlAndFile**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a931f2baa0e76732cfb2364f7ba331a32) | **(** | **out string** | **fileName** | **)** |  |

проверить Uri и файл

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **static** [**Download**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html)[**Server.Download.FromXml**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#ad75d67c74f8c87e8cf637d3197ea349f) | **(** | **XmlReader** | **reader** | **)** | **[static]** |

Загужает данные о закачке из xml файла

**Parameters:**

|  |  |
| --- | --- |
| **reader** |  |

**Returns:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **virtual void** [**Server.Download.OnDownloadCompleted**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a5efcf9fd790ad2ae716ae028a10af62b) | **(** | [**DownloadCompletedEventArgs**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_completed_event_args.html) | **e** | **)** | **[protected, virtual]** |

Событие DownloadCompleted.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **virtual void** [**Server.Download.OnDownloadProgressChanged**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a6d1d398e2c7ee312d31e4f72420b73a7) | **(** | [**DownloadProgressChangedEventArgs**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_progress_changed_event_args.html) | **e** | **)** | **[protected, virtual]** |

Событие DownloadProgressChanged event.

**Parameters:**

|  |  |
| --- | --- |
| **e** |  |
| **virtual void** [**Server.Download.OnStatusChanged**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a97782a8b48d900a97ba7ed4871da7496) | | | | **(** | **EventArgs** | **e** | **)** | **[protected, virtual]** |

Событие StatusChanged.

**Parameters:**

|  |  |
| --- | --- |
| **e** |  |
| **void** [**Server.Download.Pause**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a6a0b632dc280814176bc62dfde273ea5) | | | | **(** |  | **)** |  |

Пауза

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **void** [**Server.Download.ToXml**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a2c228786173a0ce7a7103332c0571c3c) | **(** | **XmlWriter** | **writer** | **)** |  |

Записывает данные о закачке из xml файла

**Parameters:**

|  |  |
| --- | --- |
| **reader** |  |

**Returns:**

## Property Documentation

|  |
| --- |
| **int** [**Server.Download.BufferCountPerNotification**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a8f76f908ea8c1fe0d7044a9af800fd97) **[get, set]** |

Если получим число буферов, затем возникает событие DownloadProgressChanged.

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **int** [**Server.Download.BufferSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a114b86d2e3f4628baad9b92653584d9a) **[get, set]** |

Установить BufferSize при чтении данных в поток ответа.

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **ICredentials** [**Server.Download.Credentials**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#affbbae50bbff6f235f7ebc1591eb70ff) **[get, set]** |

Сетевые учетные данные, используемые для проверки подлинности запроса

summary> Размер кэша /summary>

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **long** [**Server.Download.DownloadedSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a3506f35159b05d03c667e0577daff6c7) **[get]** |

размер загруженной чати файла

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **long** [**Server.Download.EndPoint**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#ad4b365d2703c3d38db2dac2c00f016cf) **[get, set]** |

Конечная точка

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **string** [**Server.Download.FileName**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#acbc8c865e87fed492247843a18d26158) **[get, set]** |

Имя скачиваемого файла

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **bool** [**Server.Download.HasChecked**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a4a832f971d14e7ba901734a59c83ac72) **[get, set]** |

Флаг возможности загрузки

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **bool** [**Server.Download.IsRangeSupported**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#ab33683e17870b994a91b2e09b1678442) **[get, set]** |

Укажите, следует ли удаленный сервер поддерживает "Accept-Ranges" заголовка. Используйте CheckUrl метод для инициализации этого свойства внутренне.

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **int** [**Server.Download.MaxCacheSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a90d155c9b484afd9d8af34aea00e45d9) **[get, set]** |

Размер кэша в памяти.

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **IWebProxy** [**Server.Download.Proxy**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#ae6a6197e06936067b3740dfdf2863688) **[get, set]** |

Прокси

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **long** [**Server.Download.StartPoint**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a36a2fde5cda1a65d98fb84877974b149) **[get, set]** |

Начальная точка

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **long** [**Server.Download.TotalSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a6408eb3ab98fbdc5893bb4bdeaebd422) **[get, set]** |

Размер скачиваемого файла

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **TimeSpan** [**Server.Download.TotalUsedTime**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#a0e61070551f00134e0109bda665287bb) **[get]** |

Использованное время

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

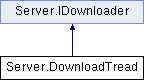
|  |
| --- |
| **Uri** [**Server.Download.Url**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html#aa7b186737ef695e09cda2accc24d0cf1) **[get, set]** |

Идентификатор ресурса

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

**Server.DownloadTread Class Reference**

Inheritance diagram for Server.DownloadTread:



|  |  |
| --- | --- |
| Public Member Functions | |
|  | [**DownloadTread**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a89dd296a3d9242834645f5afa88c021b) (string url) |
|  | Скачать весь файл |
|  | [**DownloadTread**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#af99f735af36c5698c5db93f6746e706c) (string url, long startPoint) |
|  | Скачать файл с начальной точки до конца |
|  | [**DownloadTread**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a2d6b5b7e55de50074b7555b786876cf3) (string url, long startPoint, long endPoint) |
|  | Скачать блок файла. Размер буфера по умолчанию равен 1 Кб, кэш-память является 1 Мб, а количество буферов в уведомлении 64. |
|  | **DownloadTread** (string url, long startPoint, long endPoint, int bufferSize, int cacheSize, int bufferCountPerNotification) |
| void | [**CheckUrl**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#af2d3a3cad3fa0709609df940bf45e6b9) (out string fileName) |
|  | Проверяем Uri, чтобы найти его размер, и поддерживает ли он "Пауза". |
| void | [**Download**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a6fbba4ca8cc837ceb7b312fffe33f4d2) () |
|  | Стартовать загрузку |
| void | [**BeginDownload**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#aeb7148823dd63c462f36553d5f955d24) () |
|  | Начало загрузить с помощью ThreadPool. |
| void | [**Pause**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a8d4166629112e9afd609096a96eb560f) () |
|  | Пауза загрузки |
| void | [**Resume**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a6d2264d969e1f2857864e82ec0a58162) () |
|  | Возобновить скачивание |
| void | [**BeginResume**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a6d2042a2922ecb9a7d78662b81478ca0) () |
|  | Возобновить скачивание ThreadPool. |
| void | [**Cancel**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#af6ffe1c56da9bb9b6ffa104a677d096a) () |
|  | Отмена загрузки |
| Protected Member Functions | |
| virtual void | [**OnDownloadCompleted**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a5a968935cdc24dd8536278302bcfbe8e) ([**DownloadCompletedEventArgs**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_completed_event_args.html) e) |
|  | Если загрузка завершена успешно |
| virtual void | **OnDownloadProgressChanged** ([**DownloadProgressChangedEventArgs**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_progress_changed_event_args.html) e) |
| virtual void | **OnStatusChanged** (EventArgs e) |
| Properties | |
| string | [**FileName**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#ac435c97b1a1d73ebf39b38988702ee57) [get, set] |
|  | Имя скачиваемого файла |
| Uri | [**Url**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a264d3b489e175343dd1b5bf6021bd4ab) [get, set] |
| string | [**DownloadPath**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a59dda02d3dcaa7ffc0bf7cf7d6c81abc) [get, set] |
|  | Локальный путь для сохранения файлов |
| long | [**TotalSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a5f8ff3b4bf1073b201d813a129f53d69) [get, set] |
|  | Общий размер |
| ICredentials | [**Credentials**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a82b623c30367431ed4c88516d8a6dddd) [get, set] |
|  | Сетевые учетные данные, используемые для проверки подлинности запроса |
| IWebProxy | [**Proxy**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a2ff8cc5d09925f9604d80e6da575858c) [get, set] |
| bool | [**IsRangeSupported**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a38a592cf3174f9e6c578cbf3e321d468) [get, set] |
|  | Укажите, следует ли удаленный сервер поддерживает "Accept-Ranges" заголовка. Используйте CheckUrl метод для инициализации этого свойства внутренне. |
| long | [**StartPoint**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a81c8c372c2bba224868e6b3955a94819) [get, set] |
| long | [**EndPoint**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a0b63935f18334f4a9dbf23b912564c5c) [get, set] |
| long | [**DownloadedSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a847a9d09b7801e0dd124f63d9668a6a2) [get, set] |
| int | [**CachedSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a6afd70a92356addcac4d828fe1e80f47) [get, set] |
| bool | **HasChecked** [get, set] |
| StateDownload | [**Status**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a22000af166a9d91c3c37683d032cb122) [get, set] |
|  | Состояние загрузки.Если состояние изменилось, вызываем событие StatusChanged. |
| TimeSpan | [**TotalUsedTime**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#ad65d45652545998e3ae032caef40c400) [get] |
|  | Общее время посчитанное |
| int | [**BufferCountPerNotification**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#ad0124e025476260181b1c5ff0437ba3a) [get, set] |
|  | Если получим число буферов, затем событие DownloadProgressChanged. |
| int | [**BufferSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#ae5e3056fc5a1ec3369afc14e9a21fa56) [get, set] |
|  | Установить BufferSize при чтении данных в поток ответа |
| int | [**MaxCacheSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#af911e55f0275e638a2b03f011d029109) [get, set] |
|  | Размер кэша в памяти |
| Events | |
| EventHandler < [**DownloadProgressChangedEventArgs**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_progress_changed_event_args.html) > | **DownloadProgressChanged** |
| EventHandler < [**DownloadCompletedEventArgs**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_completed_event_args.html) > | **DownloadCompleted** |
| EventHandler | **StatusChanged** |

## Constructor & Destructor Documentation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [**Server.DownloadTread.DownloadTread**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a89dd296a3d9242834645f5afa88c021b) | **(** | **string** | **url** | **)** |  |

Скачать весь файл

|  |  |  |  |
| --- | --- | --- | --- |
| [**Server.DownloadTread.DownloadTread**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a89dd296a3d9242834645f5afa88c021b) | **(** | **string** | **url,** |
|  |  | **long** | **startPoint** |
|  | **)** |  |  |

Скачать файл с начальной точки до конца

|  |  |  |  |
| --- | --- | --- | --- |
| [**Server.DownloadTread.DownloadTread**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a89dd296a3d9242834645f5afa88c021b) | **(** | **string** | **url,** |
|  |  | **long** | **startPoint,** |
|  |  | **long** | **endPoint** |
|  | **)** |  |  |

Скачать блок файла. Размер буфера по умолчанию равен 1 Кб, кэш-память является 1 Мб, а количество буферов в уведомлении 64.

## Member Function Documentation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **void** [**Server.DownloadTread.BeginDownload**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#aeb7148823dd63c462f36553d5f955d24) | **(** |  | **)** |  |

Начало загрузить с помощью ThreadPool.

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **void** [**Server.DownloadTread.BeginResume**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a6d2042a2922ecb9a7d78662b81478ca0) | **(** |  | **)** |  |

Возобновить скачивание ThreadPool.

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **void** [**Server.DownloadTread.Cancel**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#af6ffe1c56da9bb9b6ffa104a677d096a) | **(** |  | **)** |  |

Отмена загрузки

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **void** [**Server.DownloadTread.CheckUrl**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#af2d3a3cad3fa0709609df940bf45e6b9) | **(** | **out string** | **fileName** | **)** |  |

Проверяем Uri, чтобы найти его размер, и поддерживает ли он "Пауза".

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **void** [**Server.DownloadTread.Download**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a6fbba4ca8cc837ceb7b312fffe33f4d2) | **(** |  | **)** |  |

Стартовать загрузку

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **virtual void** [**Server.DownloadTread.OnDownloadCompleted**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a5a968935cdc24dd8536278302bcfbe8e) | **(** | [**DownloadCompletedEventArgs**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_completed_event_args.html) | **e** | **)** | **[protected, virtual]** |

Если загрузка завершена успешно

**Parameters:**

|  |  |
| --- | --- |
| **e** |  |
| **void** [**Server.DownloadTread.Pause**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a8d4166629112e9afd609096a96eb560f) | | | | **(** |  | **)** |  |

Пауза загрузки

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **void** [**Server.DownloadTread.Resume**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a6d2264d969e1f2857864e82ec0a58162) | **(** |  | **)** |  |

Возобновить скачивание

## Property Documentation

|  |
| --- |
| **int** [**Server.DownloadTread.BufferCountPerNotification**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#ad0124e025476260181b1c5ff0437ba3a) **[get, set]** |

Если получим число буферов, затем событие DownloadProgressChanged.

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **int** [**Server.DownloadTread.BufferSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#ae5e3056fc5a1ec3369afc14e9a21fa56) **[get, set]** |

Установить BufferSize при чтении данных в поток ответа

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **int** [**Server.DownloadTread.CachedSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a6afd70a92356addcac4d828fe1e80f47) **[get, set]** |

summary> Флаг возможности загрузки /summary>

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **ICredentials** [**Server.DownloadTread.Credentials**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a82b623c30367431ed4c88516d8a6dddd) **[get, set]** |

Сетевые учетные данные, используемые для проверки подлинности запроса

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **long** [**Server.DownloadTread.DownloadedSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a847a9d09b7801e0dd124f63d9668a6a2) **[get, set]** |

summary> Размер кэша /summary>

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **string** [**Server.DownloadTread.DownloadPath**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a59dda02d3dcaa7ffc0bf7cf7d6c81abc) **[get, set]** |

Локальный путь для сохранения файлов

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **long** [**Server.DownloadTread.EndPoint**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a0b63935f18334f4a9dbf23b912564c5c) **[get, set]** |

summary> Размер загруженных данных, который был прописан в локальном файле /summary>

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **string** [**Server.DownloadTread.FileName**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#ac435c97b1a1d73ebf39b38988702ee57) **[get, set]** |

Имя скачиваемого файла

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **bool** [**Server.DownloadTread.IsRangeSupported**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a38a592cf3174f9e6c578cbf3e321d468) **[get, set]** |

Укажите, следует ли удаленный сервер поддерживает "Accept-Ranges" заголовка. Используйте CheckUrl метод для инициализации этого свойства внутренне.

summary> Свойство начальная точка может быть использована в многопоточной загрузки сценария, и Каждый поток начинает загружать конкретный блок файла /summary>

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **int** [**Server.DownloadTread.MaxCacheSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#af911e55f0275e638a2b03f011d029109) **[get, set]** |

Размер кэша в памяти

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **IWebProxy** [**Server.DownloadTread.Proxy**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a2ff8cc5d09925f9604d80e6da575858c) **[get, set]** |

Прокси

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **long** [**Server.DownloadTread.StartPoint**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a81c8c372c2bba224868e6b3955a94819) **[get, set]** |

summary> Конечная точка /summary>

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **StateDownload** [**Server.DownloadTread.Status**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a22000af166a9d91c3c37683d032cb122) **[get, set]** |

Состояние загрузки.Если состояние изменилось, вызываем событие StatusChanged.

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **long** [**Server.DownloadTread.TotalSize**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a5f8ff3b4bf1073b201d813a129f53d69) **[get, set]** |

Общий размер

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **TimeSpan** [**Server.DownloadTread.TotalUsedTime**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#ad65d45652545998e3ae032caef40c400) **[get]** |

Общее время посчитанное

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

|  |
| --- |
| **Uri** [**Server.DownloadTread.Url**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download_tread.html#a264d3b489e175343dd1b5bf6021bd4ab) **[get, set]** |

Идентификатор ресурса

Implements [**Server.IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html).

**Server.Server Class Reference**

|  |  |
| --- | --- |
| Data Structures | |
| class | [**ConnectionInfo**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server_1_1_connection_info.html) |
|  | Информация о соединении [More...](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server_1_1_connection_info.html#details) |
| Public Member Functions | |
|  | [**Server**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a6cab4d21159f76193f63422acc97aedf) (IPEndPoint serverAddress) |
|  | Новый сервер |
| void | [**SignalCool**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a41368506576e9a663c56e0ca9aa7f3ee) ([**ConnectionInfo**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server_1_1_connection_info.html) con) |
|  | Отправить сигнал - соединение установлено |
| void | [**SignalNoCool**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a7f9139e820b6ccb8d272a09bb96cce7c) ([**ConnectionInfo**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server_1_1_connection_info.html) con) |
|  | Отправить сигнал - соединение неустановлено |
| void | [**ExitServerCmd**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#ae303cf67827875256baf5a908d446d00) () |
|  | Команда всем хостам, о том что сервер выключается |
| void | [**LoadDownloadList**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a6adccd91ae941cd6830c8aff142b0796) (string inputFile) |
|  | Загрузить список всех закачек из файла |
| Static Public Member Functions | |
| static void | [**SignalCompleted**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a47f28f2dce2a8a75c40c8935a0da0acd) (string hostName, int id, string uri, long size) |
|  | Отправить сигнал - завершена загрузка |
| static void | [**SignalFileDownload**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a1a7c687835cc43a96d666f02feb74a0f) (string hostName, int id, string uri, long size) |
|  | Отправить сигнал - загружается загрузка |
| static void | [**SignalStop**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a3141f7cab12ff8f2847053064d564663) (string hostName, int id, string uri, long size) |
|  | Отправить сигнал - стоп загрузки |
| static void | [**SignalPaused**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a4d6d6b6b22b0e7477e8ea25606bee866) (string hostName, int id, string uri, long size) |
|  | Отправить сигнал - стоп загрузки |
| static void | [**SignalNOFileDownload**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#af235f7e8541f61545594c648d08e6217) (string hostName, int id, string uri) |
|  | Отправить сигнал - невозможно загружить загрузку |
| static void | [**SignalCompleting**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a0c4d94ebf653def817febb516e9a1675) (string hostName, int id, StateDownload Status, string uri, long size, long downsize, TimeSpan time) |
|  | Отправить сигнал - невозможно загружить загрузку |
| static void | **StatusChanged** ([**Download**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html) downloader) |
| Data Fields | |
| List< [**Download**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html) > | **sf** = new List<[**Download**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_download.html)>() |

## Constructor & Destructor Documentation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [**Server.Server.Server**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a6cab4d21159f76193f63422acc97aedf) | **(** | **IPEndPoint** | **serverAddress** | **)** |  |

Новый сервер

**Parameters:**

|  |  |
| --- | --- |
| **ipEndPoint** | Структура: ip адрес и порт |

## Member Function Documentation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **void** [**Server.Server.ExitServerCmd**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#ae303cf67827875256baf5a908d446d00) | **(** |  | **)** |  |

Команда всем хостам, о том что сервер выключается

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **void** [**Server.Server.LoadDownloadList**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a6adccd91ae941cd6830c8aff142b0796) | **(** | **string** | **inputFile** | **)** |  |

Загрузить список всех закачек из файла

**Parameters:**

|  |  |
| --- | --- |
| **inputFile** |  |
| **static void** [**Server.Server.SignalCompleted**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a47f28f2dce2a8a75c40c8935a0da0acd) | | | | **(** | **string** | **hostName,** |
|  | | | |  | **int** | **id,** |
|  | | | |  | **string** | **uri,** |
|  | | | |  | **long** | **size** |
|  | | | | **)** |  | **[static]** |

Отправить сигнал - завершена загрузка

**Parameters:**

|  |  |
| --- | --- |
| **hostName** | Имя хоста |
| **id** | ID |
| **static void** [**Server.Server.SignalCompleting**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a0c4d94ebf653def817febb516e9a1675) | | | | **(** | **string** | **hostName,** |
|  | | | |  | **int** | **id,** |
|  | | | |  | **StateDownload** | **Status,** |
|  | | | |  | **string** | **uri,** |
|  | | | |  | **long** | **size,** |
|  | | | |  | **long** | **downsize,** |
|  | | | |  | **TimeSpan** | **time** |
|  | | | | **)** |  | **[static]** |

Отправить сигнал - невозможно загружить загрузку

**Parameters:**

|  |  |
| --- | --- |
| **hostName** | Имя хоста |
| **id** | ID |
| **void** [**Server.Server.SignalCool**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a41368506576e9a663c56e0ca9aa7f3ee) | | | | **(** | [**ConnectionInfo**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server_1_1_connection_info.html) | **con** | **)** |  |

Отправить сигнал - соединение установлено

**Parameters:**

|  |  |
| --- | --- |
| **con** | Соедиение |
| **static void** [**Server.Server.SignalFileDownload**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a1a7c687835cc43a96d666f02feb74a0f) | | | | **(** | **string** | **hostName,** |
|  | | | |  | **int** | **id,** |
|  | | | |  | **string** | **uri,** |
|  | | | |  | **long** | **size** |
|  | | | | **)** |  | **[static]** |

Отправить сигнал - загружается загрузка

**Parameters:**

|  |  |
| --- | --- |
| **hostName** | Имя хоста |
| **id** | ID |
| **void** [**Server.Server.SignalNoCool**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a7f9139e820b6ccb8d272a09bb96cce7c) | | | | **(** | [**ConnectionInfo**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server_1_1_connection_info.html) | **con** | **)** |  |

Отправить сигнал - соединение неустановлено

**Parameters:**

|  |  |
| --- | --- |
| **con** | Соедиение |
| **static void** [**Server.Server.SignalNOFileDownload**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#af235f7e8541f61545594c648d08e6217) | | | | **(** | **string** | **hostName,** |
|  | | | |  | **int** | **id,** |
|  | | | |  | **string** | **uri** |
|  | | | | **)** |  | **[static]** |

Отправить сигнал - невозможно загружить загрузку

**Parameters:**

|  |  |
| --- | --- |
| **hostName** | Имя хоста |
| **id** | ID |
| **static void** [**Server.Server.SignalPaused**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a4d6d6b6b22b0e7477e8ea25606bee866) | | | | **(** | **string** | **hostName,** |
|  | | | |  | **int** | **id,** |
|  | | | |  | **string** | **uri,** |
|  | | | |  | **long** | **size** |
|  | | | | **)** |  | **[static]** |

Отправить сигнал - стоп загрузки

**Parameters:**

|  |  |
| --- | --- |
| **hostName** | Имя хоста |
| **id** | ID |
| **uri** | URI |
| **static void** [**Server.Server.SignalStop**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_server.html#a3141f7cab12ff8f2847053064d564663) | | | | **(** | **string** | **hostName,** |
|  | | | |  | **int** | **id,** |
|  | | | |  | **string** | **uri,** |
|  | | | |  | **long** | **size** |
|  | | | | **)** |  | **[static]** |

Отправить сигнал - стоп загрузки

**Parameters:**

|  |  |
| --- | --- |
| **hostName** | Имя хоста |
| **id** | ID |

**Server.SaveFile Class Reference**

|  |  |
| --- | --- |
| Static Public Member Functions | |
| static HttpWebRequest | **InitializeHttpWebRequest** ([**IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html) downloader) |
| static string | [**CheckUrl**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_save_file.html#aa0d2ef717d2125fd157639bb6e051895) ([**IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html) downloader) |
|  | Проверить URL. |
| static void | [**CheckFileOrCreateFile**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_save_file.html#afca6489f055f2fed9bae5f81b921e003) ([**IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html) downloader, object fileLocker) |
|  | Проверить, существует ли файл назначения. Если нет, то создайте файл с тем же размер файла для загрузки. |

## Member Function Documentation

|  |  |  |  |
| --- | --- | --- | --- |
| **static void** [**Server.SaveFile.CheckFileOrCreateFile**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_save_file.html#afca6489f055f2fed9bae5f81b921e003) | **(** | [**IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html) | **downloader,** |
|  |  | **object** | **fileLocker** |
|  | **)** |  | **[static]** |

Проверить, существует ли файл назначения. Если нет, то создайте файл с тем же размер файла для загрузки.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **static string** [**Server.SaveFile.CheckUrl**](file:///C:\Users\Алёна\Desktop\html\class_server_1_1_save_file.html#aa0d2ef717d2125fd157639bb6e051895) | **(** | [**IDownloader**](file:///C:\Users\Алёна\Desktop\html\interface_server_1_1_i_downloader.html) | **downloader** | **)** | **[static]** |

Проверить URL.

**Parameters:**

|  |  |
| --- | --- |
| **downloader** |  |

**Returns:**

# Заключение

В результате работы над проектом были изучены возможности организации работы приложений по сети по протоколу TCP/IP на языке программирования C#. Также изучены возможности этого языка для реализации многопоточных приложений. Все основные требования к приложению были реализованы.

# Список использованных источников

1. Дейтел, Х. С# [Текст] : Учебный курс. / Дейтел Х., Дейтел П., Листфилд Дж., Нието Т., Йегер Ш., Златкина М. - СПб.: БХВ-Петербург, 2006. - 1056 с.: ил.
2. Шилдт, Г. C# [Текст] : Учебный курс. / Г. Шилдт – СПб: Питер, 2003 – 471 с.
3. Microsoft Developer Network Library [Электронный ресурс ] Режим доступа: http://msdn.microsoft.com/ru-ru/default.aspx

# Приложение А

# Руководство пользователя

Клиентская часть программы начинает работу с вывода экрана настроек (Рисунок А.1).

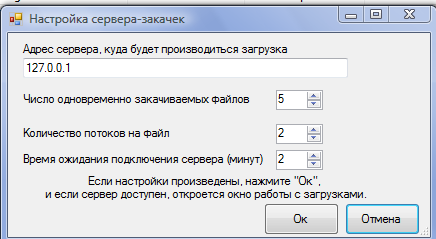


Рисунок А.1 - Меню клиента

После настройки сервера, появляется главное окно приложения. В меню предложенны три пункта (Рисунок А.2):

* Загрузка
* Стартовать
* Добавить
* Приостановить
* Остановить
* Удалить
* Настройка - служит для вызова формы настройки адреса сервера, числа одновременно закачиваемых файлов, количества потоков на файл, время ожидания подключения сервера (Рисунок А.3);

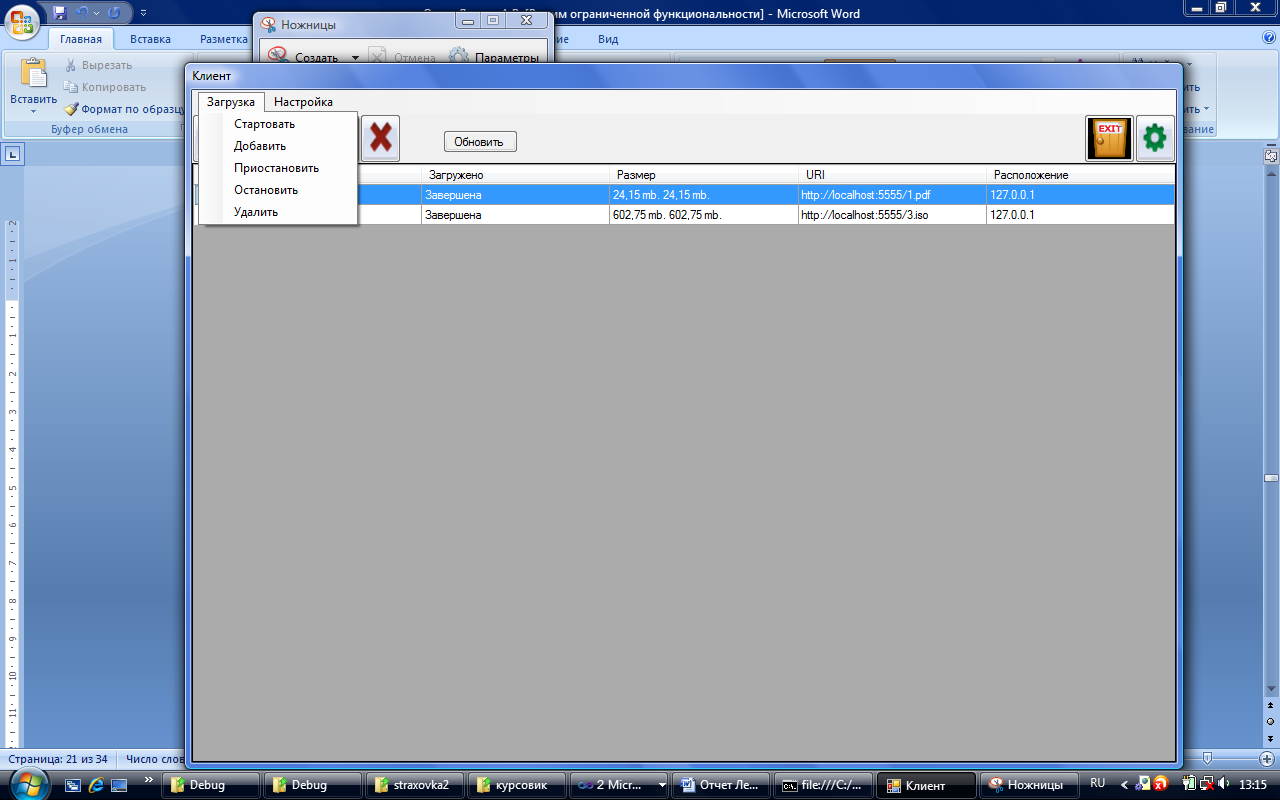


Рисунок А.2 - Меню клиента

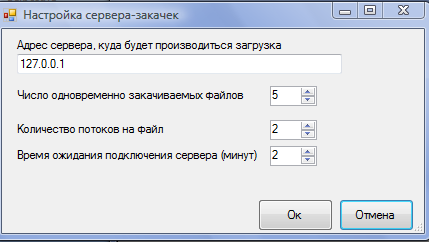


Рисунок А.3 - Настройка сервера - закачек

При нажатии кнопки добавить появляется окно, в котором требуется ввести URI загрузки (Рисунок А.4).

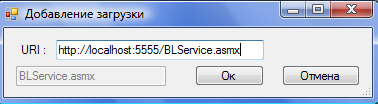


Рисунок А.4 – Добавление загрузки

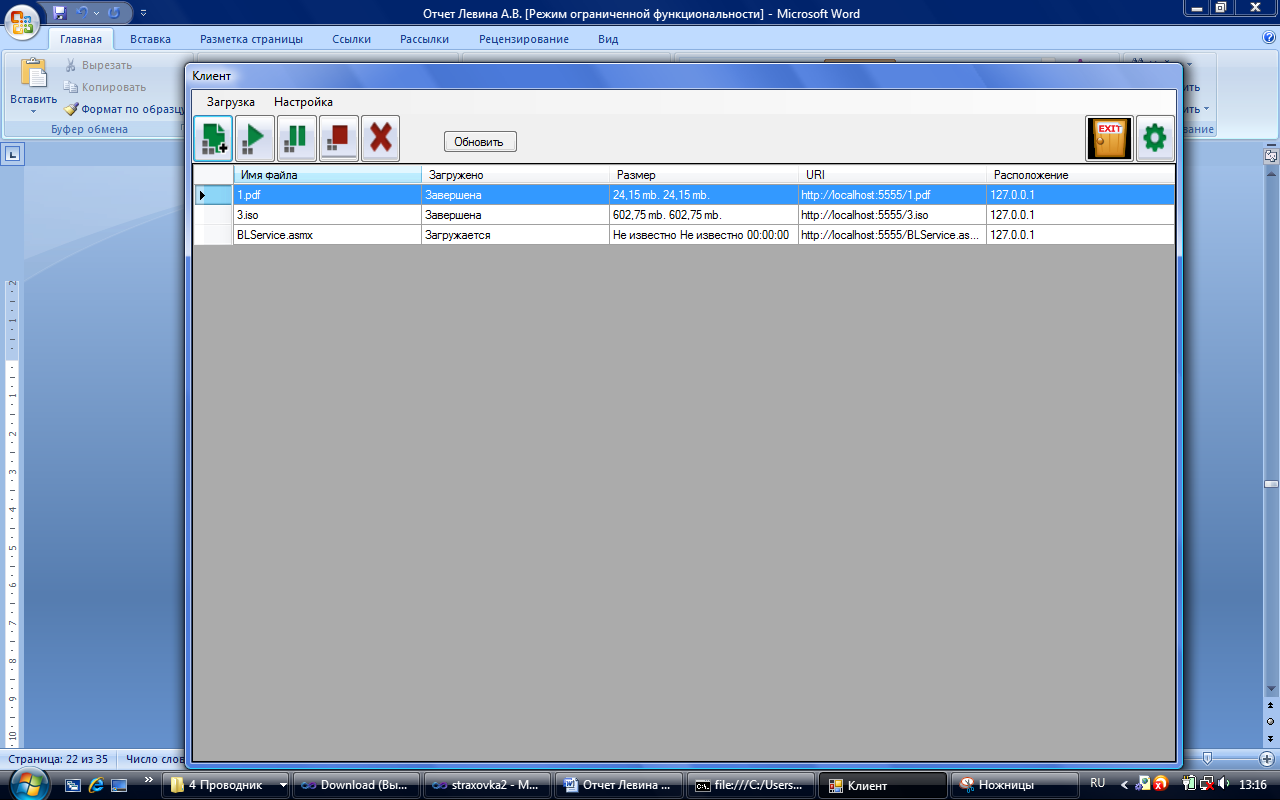


Рисунок А.5 – Отображение загрузки файла

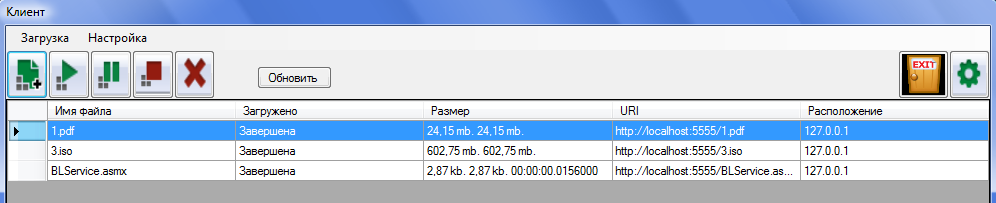


Рисунок А.6 – Отображение загрузка завершена

Серверная часть имеет вид консольного приложения (Рисунок А.7).

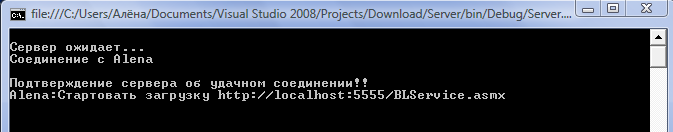


Рисунок А.7 - Главное окно серверного приложения

На сервере отображается, какие действия запросил пользователь.

# Приложение Б

# Текст программы

**Клиент**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net;

using System.Net.Sockets;

using System.Threading;

using System.IO;

using DownloadLibrary;

using System.Windows.Forms;

namespace Download

{

/// <summary>

/// Клиент

/// </summary>

public class Client : IDisposable

{

private const int INTERVAL\_CONNECT = 5000;

private TcpClient \_client;

private Thread \_clientThreaad;

private IPEndPoint \_serverAddress;

private String \_hostName;

private NetworkStream \_stream;

private BinaryReader \_reader;

private BinaryWriter \_writer;

private System.Timers.Timer \_timerConnect;

/// <summary>

/// Клиент

/// </summary>

/// <param name="serverAddress">Адрес сервера</param>

public Client(IPEndPoint serverAddress)

{

\_serverAddress = serverAddress;

\_hostName = Dns.GetHostName();

\_timerConnect = new System.Timers.Timer(Convert.ToDouble(INTERVAL\_CONNECT));

\_timerConnect.Elapsed += \_timerConnect\_Elapsed;

\_clientThreaad = new Thread(RunClient) { IsBackground = true };

\_clientThreaad.Start();

}

//Повтор соединения по таймеру

void \_timerConnect\_Elapsed(object sender, System.Timers.ElapsedEventArgs e)

{

if (\_client.Connected)

return;

try

{

\_client.Connect(\_serverAddress);

}

catch (SocketException)

{

}

}

//Запуск работы клиента (в отдельном потоке принемаем данные)

private void RunClient(object obj)

{

start:

try

{

\_client = new TcpClient();

\_timerConnect.Start();

while (true)

if (\_client.Connected)

{

\_timerConnect.Stop();

break;

}

\_stream = \_client.GetStream();

\_writer = new BinaryWriter(\_stream);

\_reader = new BinaryReader(\_stream);

string helloStr = string.Format("{0} {1}", MessageClientServer.CONNECT, \_hostName);

\_writer.Write(helloStr);

try

{

bool flag = true;

do

{

//Считываем команды из буфера

string cmd = \_reader.ReadString();

if (cmd.Contains(MessageClientServer.NODOWNLOADING)) //Cигнал-невозможно загрузить

{

string[] str = cmd.Split(' ');

string uri = str[1];

MainForm.NODOWN(uri);

}

else{

if (cmd.Contains(MessageClientServer.DOWNLOADING)) //Cигнал-загружается

{

string[] str = cmd.Split(' ');

string uri = str[1];

long size = long.Parse(str[2]);

MainForm.DOWN(uri, size);

}

else

{

if (cmd.Contains(MessageClientServer.COMPLETED)) //Cигнал-загрузка завершена

{

string[] str = cmd.Split(' ');

string uri = str[1];

long size = long.Parse(str[2]); MainForm.COMPLETED(uri,size);

}

else

{

if (cmd.Contains(MessageClientServer.PAUSE)) //Cигнал-загрузка на паузе

{

string[] str = cmd.Split(' ');

string uri = str[1];

long size = long.Parse(str[2]); MainForm.Pause(uri,size);

}

else

{

if (cmd.Contains(MessageClientServer.STOP)) //Cигнал-загрузка на стоп

{

string[] str = cmd.Split(' ');

string uri = str[1];

long size = long.Parse(str[2]);

MainForm.Stop(uri, size);

}

else

{

if (cmd.Contains(MessageClientServer.INFO)) //инфрмационные сигнал

{

string[] str = cmd.Split(' ');

string tmp = str[1]; StateDownload st=StateDownload.Nofile;

if (tmp == "Downloading")

st = StateDownload.Downloading;

else

{

if (tmp == "Completed")

st = StateDownload.Completed;

else

if (tmp == "Paused")

st = StateDownload.Paused;

else

if (tmp == "Stopped")

st = StateDownload.Stopped;

else

if (tmp == "Created")

st = StateDownload.Created;

}

string uri = str[2];

long size = long.Parse(str[3]);

long downsize = long.Parse(str[4]);

TimeSpan time = TimeSpan.Parse(str[5]); MainForm.Information(st,uri, size,downsize,time);

}

else

{

switch (cmd)

{

//соединение не установлено

case MessageClientServer.NOCOOL: break;

//соединение установлено

case MessageClientServer.COOL:

MainForm.flagConnect = false;

break;

//Выключение сервера

case MessageClientServer.EXIT\_SERVER:

{

MessageBox.Show("Сервер отключился.");

CloseResourse();

flag = false;

break;

}

}

}

}

}

}

}

}

}while (flag);

}

catch (SocketException)

{

MessageBox.Show("Соединение было не ожиданно разорванно!");

CloseResourse();

goto start;

}

}

catch (Exception exp)

{

CloseResourse();

MessageBox.Show(exp.Message);

}

finally

{

CloseResourse();

}

}

/// <summary>

/// Отправка на сервер сигнала на загрузку файла

/// </summary>

public void SignalServerDownload(string strUri)

{

\_writer.Write(MessageClientServer.START+" "+strUri+" "+Properties.Settings.Default.numTread.ToString());

}

/// <summary>

/// Отправка на сервер сигнала на загрузку имеющегося файла

/// </summary>

public void SignalServerDownloadLast(string strUri)

{

\_writer.Write(MessageClientServer.LAST + " " + strUri);

}

/// <summary>

/// Отправка на сервер сигнала на установку на паузу загрузку файла

/// </summary>

public void SignalServerPause(string strUri)

{

\_writer.Write(MessageClientServer.PAUSE + " " + strUri);

}

/// <summary>

/// Отправка на сервер сигнала остановки загрузки файла

/// </summary>

public void SignalServerStop(string strUri)

{

\_writer.Write(MessageClientServer.STOP + " " + strUri);

}

/// <summary>

/// Отправка на сервер сигнала удаление загрузки файла

/// </summary>

public void SignalServerDelete(string strUri)

{

\_writer.Write(MessageClientServer.DELETE + " " + strUri);

}

/// <summary>

/// Отправка на сервер команды об отключении клиента

/// </summary>

public void CloseClientCmd()

{

if (\_client != null && \_client.Connected)

\_writer.Write(MessageClientServer.CLOSE\_CONNECT);

}

//Закрытие ресурсов

private void CloseResourse()

{

//Закрытие соединения

\_writer.Close();

\_reader.Close();

\_stream.Close();

\_client.Close();

}

//Уничтожение объекта

public void Dispose()

{

Dispose(true);

GC.SuppressFinalize(this);

}

//Освобождение памяти

protected virtual void Dispose(bool disposing)

{

if (disposing)

{

if (\_stream != null)

{

\_stream.Dispose();

\_stream = null;

}

if (\_reader != null)

{

\_reader.Dispose();

\_reader = null;

}

if (\_writer != null)

{

\_writer.Dispose();

\_writer = null;

}

if (\_timerConnect != null)

{

\_timerConnect.Dispose();

\_timerConnect = null;

}

}

}

//Деструтор

~Client()

{

Dispose(false);

}

}

}

**Сервер**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net;

using System.Net.Sockets;

using System.Threading;

using System.IO;

using DownloadLibrary;

using System.Xml;

namespace Server

{

public class Server

{

private Thread \_serverThread;

private IPEndPoint \_serverAddress;

private TcpListener \_serverListener;

static private List<ConnectionInfo> \_connections;

private int \_maxConnectCount;

public List<Download> sf = new List<Download>();

WebProxy proxy = null;

/// <summary>

/// Информация о соединении

/// </summary>

public class ConnectionInfo : IDisposable

{

public Socket Socket;

public Thread Thread;

public String HostName;

public NetworkStream Stream;

public BinaryWriter Writer;

public BinaryReader Reader;

public int id;

//public StateConputer state;

public bool isPlay;

/// <summary>

/// Новое соединение

/// </summary>

/// <param name="socket">Сокет</param>

/// <param name="id">ID</param>

public ConnectionInfo(Socket socket, int id)

{

this.Socket = socket;

this.Stream = new NetworkStream(this.Socket);

this.Reader = new BinaryReader(this.Stream);

this.Writer = new BinaryWriter(this.Stream);

this.id = id;

this.isPlay = false;

}

/// <summary>

/// Строковое представление

/// </summary>

/// <returns></returns>

public override string ToString()

{

if (!string.IsNullOrWhiteSpace(HostName))

return String.Format("{0}", HostName);

return "Определяется имя хоста...";

}

/// <summary>

/// Уничтожение объекта

/// </summary>

public void Dispose()

{

Dispose(true);

GC.SuppressFinalize(this);

}

//Освобожение ресурсов

protected virtual void Dispose(bool disposing)

{

if (disposing)

{

if (Socket != null)

{

Socket.Dispose();

Socket = null;

}

if (Stream != null)

{

Stream.Dispose();

Stream = null;

}

if (Writer != null)

{

Writer.Dispose();

Writer = null;

}

if (Reader != null)

{

Reader.Dispose();

Reader = null;

}

}

}

//Деструктор

~ConnectionInfo()

{

Dispose(false);

}

}

/// <summary>

/// Новый сервер

/// </summary>

/// <param name="ipEndPoint">Структура: ip адрес и порт</param>

public Server(IPEndPoint serverAddress)

{

\_serverAddress = serverAddress;

\_connections = new List<ConnectionInfo>();

\_serverListener = new TcpListener(\_serverAddress);

\_serverThread = new Thread(AcceptConnections) { IsBackground = true };

\_serverThread.Start();

}

//Принимаем новые соединения (обработку каждого помещаем в отдельный поток)

private void AcceptConnections()

{

\_serverListener.Start();

while (true)

{

Socket socket = \_serverListener.AcceptSocket();

\_maxConnectCount++;

ConnectionInfo connection = new ConnectionInfo(socket, \_maxConnectCount)

{

Thread = new Thread(ProcessConnection) { IsBackground = true }

};

connection.Thread.Start(connection);

lock (\_connections)

\_connections.Add(connection);

}

}

//Обработка отдельного соединения

private void ProcessConnection(object state)

{

ConnectionInfo connection = (ConnectionInfo)state;

try

{

bool flag = true;

do

{

string cmd = connection.Reader.ReadString();

if (cmd.Contains(MessageClientServer.CONNECT))

{

try

{

string[] str = cmd.Split(' ');

connection.HostName = str[1];

Console.WriteLine("Соединение с {0} \n", connection.HostName);

SignalCool(connection);

}catch(Exception er)

{

Console.WriteLine(er.ToString());

SignalNoCool(connection);

}

}

else

{

if (cmd.Contains(MessageClientServer.START)) //Cигнал на загрузку файла

{

string[] str = cmd.Split(' ');

string uri = str[1];

int maxThread = Convert.ToInt32(str[2]);

Download d = new Download(uri,maxThread);

d.Proxy = this.proxy;

string filename = string.Empty;

d.FileName=MessageClientServer.GenerateFileNameFromUri(uri);

d.HostName = connection.HostName;

d.id = connection.id;

d.DownloadPath = d.DownloadPath + @"\"+d.FileName;

d.DownloadCompleted += DownloadCompleted;

d.DownloadProgressChanged += DownloadProgressChanged;

d.StatusChanged += StatusChanging;

d.BeginDownload();

sf.Add(d);

Console.WriteLine("{0}:Стартовать загрузку {1} \n", connection.HostName,uri);

StatusChanged(d);

}

else

{

if (cmd.Contains(MessageClientServer.LAST)) //Cигнал на загрузку имеющегося файла

{

string[] str = cmd.Split(' ');

string uri = str[1];

for (int i = 0; i < sf.Count; i++)

{

if (sf[i].Url.AbsoluteUri == uri)

{

sf[i].BeginResume();

Console.WriteLine("{0}:Стартовать имеющуюся загрузку {1} \n", connection.HostName, uri);

StatusChanged(sf[i]);

}

}

}

else

{

if (cmd.Contains(MessageClientServer.STOP)) //Стоп загрузки

{

string[] str = cmd.Split(' ');

string uri = str[1];

for (int i = 0; i < sf.Count; i++)

{

if (sf[i].Url.AbsoluteUri == uri)

{

sf[i].Cancel();

Console.WriteLine("{0}:Остановить загрузку {1} \n", connection.HostName, uri);

StatusChanged(sf[i]);

}

}

}

else

{ if(cmd.Contains(MessageClientServer.PAUSE)) //Пауза загрузки

{

string[] str = cmd.Split(' ');

string uri = str[1];

for (int i = 0; i < sf.Count; i++)

{

if (sf[i].Url.AbsoluteUri == uri)

{

sf[i].Pause();

Console.WriteLine("{0}:Приостановить загрузку {1} \n", connection.HostName, uri);

StatusChanged(sf[i]);

}

}

}

else

{

if (cmd.Contains(MessageClientServer.DELETE)) //Удаление

{

string[] str = cmd.Split(' ');

string uri = str[1];

int[] tmp = new int[sf.Count];

int k = 0;

for (int i = 0; i < tmp.Length; i++)

tmp[i] = -1;

for (int i = 0; i < sf.Count; i++)

{

if (sf[i].Url.AbsoluteUri == uri && sf[i].HostName == connection.HostName)

{

try

{

if (File.Exists(sf[i].DownloadPath))

File.Delete(sf[i].DownloadPath);

}

catch (Exception ex)

{

Console.WriteLine("Невозможно удалить файл | " + ex.Message);

}

// sf[i].Cancel();

Console.WriteLine("{0}:Удалить загрузку {1} \n", connection.HostName, uri);

tmp[k] = i; k++;

}

}

if (tmp != null)

{

for (int i = tmp.Length - 1; i >= 0; i--)

{

if (tmp[i] != -1)

{

sf.RemoveAt(tmp[i]);

}

}

}

}

else

{

switch (cmd)

{

//Хост разрывает соединение

case MessageClientServer.CLOSE\_CONNECT:

{

Console.WriteLine("Клиент {0} отключился.", connection.HostName);

flag = false;

break;

} }

}

}

}

}

}

}

} while (flag && connection.Socket.Connected);

}

catch (SocketException exp)

{

Console.WriteLine(string.Format("SocketException на хосте: {0} - {1}", connection.HostName, exp.Message));

}

catch (Exception exp)

{

Console.WriteLine(string.Format("Exception на хосте: {0} - {1}", connection.HostName, exp.Message));

}

finally

{

connection.Writer.Close();

connection.Reader.Close();

connection.Stream.Close();

connection.Socket.Close();

lock (\_connections)

\_connections.Remove(connection);

SaveDownloadList("download.xml");

}

}

//Поиск соединения по id и номеру хоста

private static ConnectionInfo SearchConnection(string hostName, int id)

{

lock (\_connections)

{

foreach (ConnectionInfo c in \_connections)

if (c.id == id && c.HostName == hostName)

return c;

return null;

}

}

/// <summary>

/// Отправить сигнал - соединение установлено

/// </summary>

/// <param name="con">Соедиение</param>

public void SignalCool(ConnectionInfo con)

{

con.Writer.Write(MessageClientServer.COOL); Console.WriteLine("Подтверждение сервера об удачном соединении!!"); }

/// <summary>

/// Отправить сигнал - соединение неустановлено

/// </summary>

/// <param name="con">Соедиение</param>

public void SignalNoCool(ConnectionInfo con)

{

con.Writer.Write(MessageClientServer.NOCOOL);

Console.WriteLine("Подтверждение сервера об неудачном соединении!!");

}

/// <summary>

/// Отправить сигнал - завершена загрузка

/// </summary>

/// <param name="hostName">Имя хоста</param>

/// <param name="id">ID</param>

public static void SignalCompleted(string hostName, int id, string uri, long size)

{

ConnectionInfo connection = SearchConnection(hostName, id);

//Не нашли на выход

if (connection == null)

return;

//Если нашли, отправим соответствующую команду серверу

connection.Writer.Write(MessageClientServer.COMPLETED + " " + uri+" "+size);

}

/// <summary>

/// Отправить сигнал - загружается загрузка

/// </summary>

/// <param name="hostName">Имя хоста</param>

/// <param name="id">ID</param>

public static void SignalFileDownload(string hostName, int id, string uri, long size)

{

ConnectionInfo connection = SearchConnection(hostName, id);

//Не нашли на выход

if (connection == null)

return;

//Если нашли, отправим соответствующую команду серверу

connection.Writer.Write(MessageClientServer.DOWNLOADING+" "+uri+" "+size);

}

/// <summary>

/// Отправить сигнал - стоп загрузки

/// </summary>

/// <param name="hostName">Имя хоста</param>

/// <param name="id">ID</param>

public static void SignalStop(string hostName, int id, string uri, long size)

{

ConnectionInfo connection = SearchConnection(hostName, id);

//Не нашли на выход

if (connection == null)

return;

//Если нашли, отправим соответствующую команду серверу

connection.Writer.Write(MessageClientServer.STOP+" "+uri+" "+size);

}

/// <summary>

/// Отправить сигнал - стоп загрузки

/// </summary>

/// <param name="hostName">Имя хоста</param>

/// <param name="id">ID</param>

/// <param name="uri">URI</param>

public static void SignalPaused(string hostName, int id, string uri, long size)

{

ConnectionInfo connection = SearchConnection(hostName, id);

//Не нашли на выход

if (connection == null)

return;

//Если нашли, отправим соответствующую команду серверу

connection.Writer.Write(MessageClientServer.PAUSE+" "+uri+" "+size);

}

/// <summary>

/// Отправить сигнал - невозможно загружить загрузку

/// </summary>

/// <param name="hostName">Имя хоста</param>

/// <param name="id">ID</param>

public static void SignalNOFileDownload(string hostName, int id, string uri)

{

ConnectionInfo connection = SearchConnection(hostName, id);

//Не нашли на выход

if (connection == null)

return;

//Если нашли, отправим соответствующую команду серверу

connection.Writer.Write(MessageClientServer.NODOWNLOADING + " " + uri);

}

/// <summary>

/// Отправить сигнал - невозможно загружить загрузку

/// </summary>

/// <param name="hostName">Имя хоста</param>

/// <param name="id">ID</param>

public static void SignalCompleting(string hostName, int id, StateDownload Status, string uri, long size, long downsize,TimeSpan time)

{

ConnectionInfo connection = SearchConnection(hostName, id);

//Не нашли на выход

if (connection == null)

return;

//Если нашли, отправим соответствующую команду серверу

connection.Writer.Write(MessageClientServer.INFO + " " + Status + " " + uri + " " + size + " " + downsize + " " + time.ToString());

}

/// <summary>

/// Команда всем хостам, о том что сервер выключается

/// </summary>

public void ExitServerCmd()

{

lock (\_connections)

{

foreach (ConnectionInfo c in \_connections)

c.Writer.Write(MessageClientServer.EXIT\_SERVER);

\_connections.Clear();

SaveDownloadList("download.xml");

}

}

/// <summary>

/// Сохранить список всех закачек в файл

/// </summary>

/// <param name="outputFile"></param>

private void SaveDownloadList(string outputFile)

{

XmlWriter writer = null;

try

{

XmlWriterSettings settings = new XmlWriterSettings()

{

Indent = true,

};

writer = XmlWriter.Create(File.Create(outputFile), settings);

writer.WriteStartElement("DownloadsArray");

for (int i = 0; i < sf.Count; i++)

{

Download current = sf[i];

current.ToXml(writer);

}

writer.WriteEndElement();

}

catch (Exception ex)

{

Console.WriteLine("Ошибка при записи списка закачек в файл | " + ex.Message);

}

finally

{

if (writer != null) writer.Close();

}

}

public static void StatusChanged(Download downloader)

{

switch (downloader.Status)

{

case StateDownload.Downloading: SignalFileDownload(downloader.HostName, downloader.id, downloader.Url.AbsoluteUri, downloader.TotalSize);

break;

case StateDownload.Stopped: SignalStop(downloader.HostName, downloader.id, downloader.Url.AbsoluteUri,downloader.TotalSize);

break;

case StateDownload.Completed: SignalCompleted(downloader.HostName, downloader.id, downloader.Url.AbsoluteUri, downloader.TotalSize);

break;

case StateDownload.Paused: SignalPaused(downloader.HostName, downloader.id, downloader.Url.AbsoluteUri, downloader.TotalSize);

break;

case StateDownload.Nofile: SignalNOFileDownload(downloader.HostName, downloader.id, downloader.Url.AbsoluteUri);

break;

}

if (downloader.Status == StateDownload.Paused)

{

string tmp=

String.Format("Отправлено: {0}KB, Общий размер: {1}KB, Время: {2}:{3}:{4}",

downloader.DownloadedSize / 1024, downloader.TotalSize / 1024,

downloader.TotalUsedTime.Hours, downloader.TotalUsedTime.Minutes,

downloader.TotalUsedTime.Seconds);

Console.WriteLine(tmp+"\n");

}

}

/// <summary>

/// Загрузить список всех закачек из файла

/// </summary>

/// <param name="inputFile"></param>

public void LoadDownloadList(string inputFile)

{

XmlReader reader = null;

try

{

XmlReaderSettings settings = new XmlReaderSettings();

settings.ConformanceLevel = ConformanceLevel.Fragment;

settings.IgnoreWhitespace = true;

settings.IgnoreComments = true;

settings.CloseInput = true;

reader = XmlReader.Create(File.OpenRead(inputFile), settings);

reader.Read();

if (reader.NodeType != XmlNodeType.XmlDeclaration) throw new FormatException();

reader.ReadStartElement("DownloadsArray");

if (reader.ReadState == ReadState.EndOfFile)

return;

while ((reader.Name != "DownloadsArray") && (reader.NodeType != XmlNodeType.EndElement))

{

Download d = Download.FromXml(reader);

sf.Add(d);

reader.Read();

}

}

catch (Exception ex)

{

Console.WriteLine("Ошибка при первоначальной загрузке списка закачек: " + ex.Message);

}

finally

{

if (reader != null)

{

reader.Close();

}

}

}

void DownloadCompleted(object sender, DownloadCompletedEventArgs e)

{

if (sf.Count >= 1)

SignalCompleting(sf[sf.Count - 1].HostName, sf[sf.Count - 1].id, sf[sf.Count-1].Status, sf[sf.Count - 1].Url.AbsoluteUri,sf[sf.Count - 1].TotalSize, e.DownloadedSize,e.TotalTime);

}

void DownloadProgressChanged(object sender, DownloadProgressChangedEventArgs e)

{

if (sf.Count >= 1)

SignalCompleting(sf[sf.Count - 1].HostName, sf[sf.Count - 1].id, sf[sf.Count - 1].Status, sf[sf.Count - 1].Url.AbsoluteUri, sf[sf.Count - 1].TotalSize, sf[sf.Count - 1].DownloadedSize, sf[sf.Count - 1].TotalUsedTime);

}

void StatusChanging(object sender, EventArgs e)

{

if (sf.Count >= 1)

StatusChanged(sf[sf.Count - 1]);

}

}

}