Лабораторная работа №2
Работа с FTP-сервером

Теоретическая часть

FTP (англ. File Transfer Protocol — протокол передачи файлов) — стандартный протокол, предназначенный для передачи файлов по TCP-сетям (например, Интернет). FTP часто используется для загрузки сетевых страниц и других документов с частного устройства разработки на открытые сервера хостинга.

Протокол построен на архитектуре «клиент-сервер» и использует разные сетевые соединения для передачи команд и данных между клиентом и сервером. Пользователи FTP могут пройти аутентификацию, передавая логин и пароль открытым текстом, или же, если это разрешено на сервере, они могут подключиться анонимно. Можно использовать протокол SSH для безопасной передачи, скрывающей (шифрующей) логин и пароль, а также шифрующей содержимое.

Первые клиентские FTP-приложения были интерактивными инструментами командной строки, реализующими стандартные команды и синтаксис. Графические пользовательские интерфейсы с тех пор были разработаны для многих используемых по сей день операционных систем. Среди этих интерфейсов как программы общего веб-дизайна вроде Microsoft Expression Web, так и специализированные FTP-клиенты (например, FileZilla).

Для разработки сетевого приложения-клиента будем использовать бесплатную библиотеку BytesRoad.NetSuit.

В коплекте 3 библиотеки:

BytesRoad.Diag.dll

BytesRoad.Net.Ftp.dll

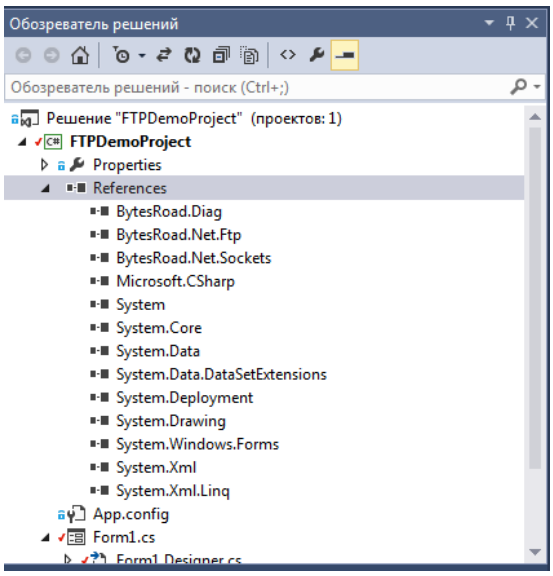
BytesRoad.Net.Sockets.dll

Практическая часть

Для примера создадим приложение-клиент, которое подключается и скачивает определенный файл с сервера.

Создадим новый проект Приложение Windows Forms на языке C#.

Подключим библиотеки для работы с ftp



Весь код подключения и скачивания файла разместим в обработчике события нажатия на кнопку. Создадим экземпляр класса FtpClient и сразу укажем, что будем работать в пассивном режиме:

```
FtpClient client = new FtpClient();
client.PassiveMode = true;
```

Основные параметры подключения укажем в специальных переменных:

```
int TimeoutFTP = 30000; //Таймаут.
string FTP_SERVER = "localhost";
int FTP_PORT = 953;
string FTP_USER = "student";
string FTP_PASSWORD = "student";
```

Если FTP сервер находится за прокси-сервером тогда необходимо указать:

```
FtpProxyInfo pinfo = new FtpProxyInfo(); //Это переменная параметров.
pinfo.Server = "192.168.1.240";
pinfo.Port = 3128; //Порт.
pinfo.Type = FtpProxyType.HttpConnect; //Тип прокси - всего 4 вида.
pinfo.PreAuthenticate = true; //Если на прокси есть идентификация
pinfo.User = "student";
pinfo.Password = "student";

//Присваиваем параметры прокси клиенту.
client.ProxyInfo = pinfo;
```

Настройка клиента закончена. Далее необходимо подключится. По правилам хорошего программиста заключим код подключения в блок try-

catch и перехватим исключение типа System.Net.Sockets.SocketException и BytesRoad.Net.Ftp.FtpTimeoutException:

```
//Подключаемся к FTP серверу.
try
{
    client.Connect(TimeoutFTP, FTP_SERVER, FTP_PORT);
}
catch(BytesRoad.Net.Ftp.FtpTimeoutException error)
{
    MessageBox.Show("Время ожидания истекло! Сервер не отвечает. " + error.Message,
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    return;
}
catch (System.Net.Sockets.SocketException error)
{
    MessageBox.Show(error.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    return;
} }
```

В случае успешного подключения необходимо авторизоваться на сервер (не забыв о перехвате исключений!!!):

```
try
{
    client.Login(TimeoutFTP, FTP_USER, FTP_PASSWORD);
}
catch (BytesRoad.Net.Ftp.FtpTimeoutException error)
{
    MessageBox.Show("Время ожидания истекло! Сервер не отвечает. " + error.Message,
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    return;
}
catch (System.Net.Sockets.SocketException error)
{
    MessageBox.Show(error.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    return;
}
```

Затем получим список файлов в корневой директории. Файл представляет собой структуру FtpItem:

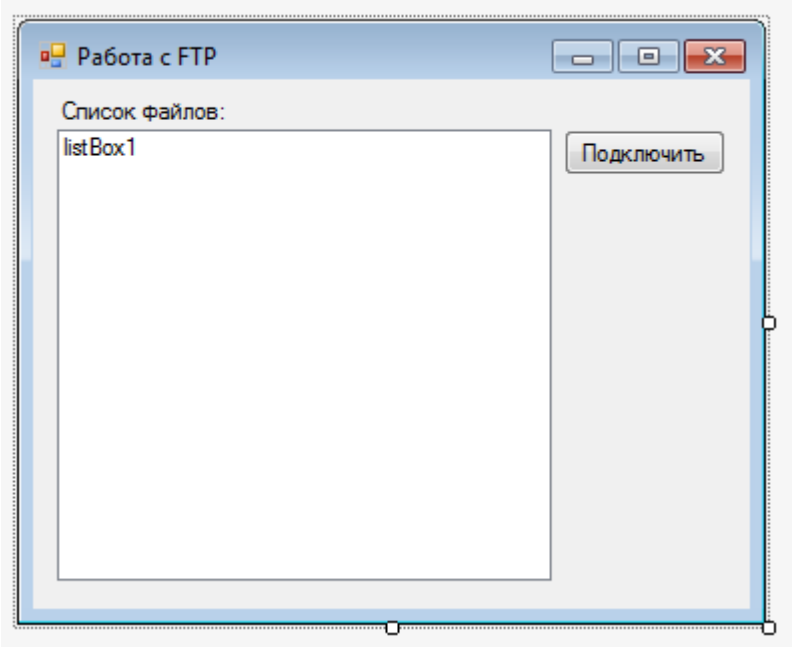
```
    FtpItem[] dirs = client.GetDirectoryList(TimeoutFTP);
```

Сохраним первый файл (если он есть) на текущий компьютер, причем место сохранения получим с помощью стандартного диалога сохранить:

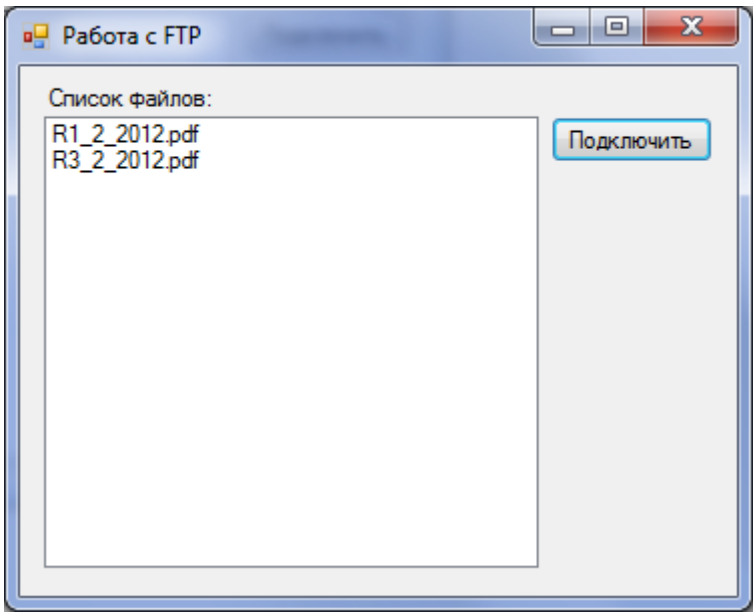
```
foreach (FtpItem dir in dirs)
{
    listBox1.Items.Add(dir.Name);
}

if (dirs.Count() > 0)
{
    saveFileDialog1.FileName = dirs[0].Name;
    if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)
    {
```

```
            client.GetFile(TimeoutFTP, saveFileDialog1.FileName, dirs[0].Name);
        }
}
```

Форма приложения получилась очень простой:



Результат работы:



Остальные функции для работы с FTP:

Теперь немного о самых основных методах для работы с FTP сервером:

```
//Получает список содержимого текущего каталога с FTP.
client.GetDirectoryList(TimeoutFTP);

//Меняет директорию на указанную.
//Можно переходить вверх указав вместо имени папки ".." либо в любую папку
расположенную в текущей.
```

```
client.ChangeDirectory(TimeoutFTP, "папка");

//Удаляет указанный файл с сервера.
client.DeleteFile(TimeoutFTP, "файл");

//Удаляет указанную папку с сервера.
client.DeleteDirectory(TimeoutFTP, "файл");

//Принимает указанный файл с сервера.
client.GetFile(TimeoutFTP, "куда принимаем - путь на диске", "Что принимаем -
файл на сервере");

//загружаем файл на сервер.
client.PutFile(TimeoutFTP, "имя файла на сервере", "что грузим - имя файла на
компьютере");
```

Подробности в файле справки к библиотеке BytesRoad.NetSuit_Ref.chm.

Задание на лабораторную работу

Разработать FTP-клиент, осуществляющий:

1. поиск файла по имени на FTP-сервере.
2. замену файла с указанным именем на указанный файл.
3. выкачивание содержимого указанного каталога с сервера (с сохранением подкаталогов).
4. закачивание содержимого указанного каталога на сервер (с сохранением подкаталогов).
5. копирование структуры каталогов в указанном месте на диске.
6. удаление файлов по указанной маске.
7. воссоздание структуры каталогов на сервере по подобию указанного места на диске.
8. сравнение указанного каталога на диске с каталогов на сервере.
9. синхронизация указанного каталога на диске с каталогов на сервере, путем замены более старых файлов новыми (без создания новых файлов).
10. синхронизация указанного каталога на диске с каталогом на сервере (без обновления существующих).
11. отбор файлов по дате создания (за неделю, месяц и т.п.).
12. работу с командной строкой (стандартные команды ftp протокола).
13. подстчет размеров содержимого папок и последнюю дату изменения.
14. вывод статистики по FTP-серверу (объем, количество файлов каждого расширения, количество папок, средний размер и количество файлов в папках др.).