

# Re: FTP Upload

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.aspnet/2008-01/msg02223.html>

---

- *From:* "MDB" <mdb@xxxxxxxxxxx>
  - *Date:* Tue, 29 Jan 2008 21:14:39 -0500
- 

Thanks

"Misbah Arefin" <MisbahArefin@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message  
<news:FF052CB4-972C-4894-9C5A-EB65836CE3C8@xxxxxxxxxxxxxxxxxxxx>

i got this from some site a couple of years back (dont recall which site)

```
////////////////////
using System;
using System.Net;
using System.IO;
using System.Text;
using System.Net.Sockets;

namespace SampleCode
{
    public class clsFTP
    {
        #region "Class Variable Declarations"
        private string m_sRemoteHost, m_sRemotePath, m_sRemoteUser;
        private string m_sRemotePassword, m_sMess;
        private int m_iRemotePort, m_iBytes;
        private Socket m_objClientSocket;

        private int m_iRetVal;
        private bool m_bLoggedIn;
        private string m_sMes, m_sReply;

        //set the size of the packet that is used to read and to write data to the
        //FTP server to the following specified size.
        public const int BLOCK_SIZE = 512;
        private byte[] m_aBuffer = new byte[BLOCK_SIZE];
        private Encoding ASCII = Encoding.ASCII;
        public bool flag_bool;
        //general variable declaration
        private string m_sMessageString;
        #endregion
    }
}
```

```
#region "Class Constructors"

//main class constructor
public clsFTP()
{
    m_sRemoteHost = "microsoft";
    m_sRemotePath = ".";
    m_sRemoteUser = "anonymous";
    m_sRemotePassword = "";
    m_sMessageString = "";
    m_iRemotePort = 21;
    m_bLoggedIn = false;
}

//parameterized constructor
public clsFTP(string sRemoteHost, string sRemotePath, string sRemoteUser,
string sRemotePassword, int iRemotePort)
{
    m_sRemoteHost = sRemoteHost;
    m_sRemotePath = sRemotePath;
    m_sRemoteUser = sRemoteUser;
    m_sRemotePassword = sRemotePassword;
    m_sMessageString = "";
    m_iRemotePort = 21;
    m_bLoggedIn = false;
}
#endregion

#region "Public Properties"

//set or get the name of the FTP server that you want to connect.
public string RemoteHostFTPServer
{
    //get the name of the FTP server.
    get{return m_sRemoteHost;}
    //set the name of the FTP server.
    set{m_sRemoteHost = value;}
}

//set or get the FTP Port Number of the FTP server that you want to connect.
public int RemotePort
{
    //get the FTP Port Number.
    get{return m_iRemotePort;}
    //set the FTP Port Number.
    set{m_iRemotePort = value;}
}

//set or get the remote path of the FTP server that you want to connect.
public string RemotePath
{
```

## Re: FTP Upload

```
//get the remote path.
get{return m_sRemotePath;}
//set the remote path.
set{m_sRemotePath = value;}
}

//set or get the remote password of the FTP server that you want to connect.
public string RemotePassword
{
get{return m_sRemotePassword;}
set{m_sRemotePassword = value;}
}

//set or get the remote user of the FTP server that you want to connect.
public string RemoteUser
{
get{return m_sRemoteUser;}
set{m_sRemoteUser = value;}
}

//set the class MessageString.
public string MessageString
{
get{return m_sMessageString;}
set{m_sMessageString = value;}
}
#endregion

#region "Public Subs and Functions"

//return a list of files in a string array from the file system.
public string[] GetFileList(string sMask)
{
Socket cSocket;
int bytes;
char seperator = '\n';
string[] mess;

m_sMes = "";
//check if you are logged on to the FTP server.
if(!(m_bLoggedIn))
{
Login();
}

cSocket = CreateDataSocket();
//send an FTP command,
SendCommand("NLST " + sMask);

if(!(m_iRetVal == 150 || m_iRetVal == 125))
{
```

## Re: FTP Upload

```
MessageString = m_sReply;
throw new IOException(m_sReply.Substring(4));
}

m_sMes = "";
while(true)
{
Array.Clear(m_aBuffer, 0, m_aBuffer.Length);
bytes = cSocket.Receive(m_aBuffer, m_aBuffer.Length, 0);
m_sMes += ASCII.GetString(m_aBuffer, 0, bytes);

if(bytes < m_aBuffer.Length)
{
break;
}
}

mess = m_sMes.Split(seperator);
cSocket.Close();
ReadReply();

if(m_iRetVal != 226)
{
MessageString = m_sReply;
throw new IOException(m_sReply.Substring(4));
}

return mess;
}

//get the size of the file on the FTP server.
public long GetFileSize(string sFileName)
{
long size;

if(!(m_bLoggedIn))
{
Login();
}
//send an FTP command.
SendCommand("SIZE " + sFileName);
size = 0;

if(m_iRetVal == 213)
{
size = long.Parse(m_sReply.Substring(4));
}
else
{
MessageString = m_sReply;
throw new IOException(m_sReply.Substring(4));
}
```

## Re: FTP Upload

```
}

return size;
}

//log on to the FTP server.
public bool Login()
{
    m_objClientSocket = new Socket(AddressFamily.InterNetwork,
    SocketType.Stream, ProtocolType.Tcp);

    IPEndPoint ep = new IPEndPoint(Dns.Resolve(m_sRemoteHost).AddressList[0],
    m_iRemotePort);

    try
    {
        m_objClientSocket.Connect(ep);
    }
    catch(Exception ex)
    {
        MessageString = m_sReply;
        throw new IOException("Cannot connect to remote server");
    }

    ReadReply();
    if(m_iRetValue != 220)
    {
        CloseConnection();
        MessageString = m_sReply;
        throw new IOException(m_sReply.Substring(4));
    }
    //send an FTP command to send a user logon ID to the server.
    SendCommand("USER " + m_sRemoteUser);
    if(!(m_iRetValue == 331 || m_iRetValue == 230))
    {
        Cleanup();
        MessageString = m_sReply;
        throw new IOException(m_sReply.Substring(4));
    }

    if(m_iRetValue != 230)
    {
        //send an FTP command to send a user logon password to the server.
        SendCommand("PASS " + m_sRemotePassword);
        if(!(m_iRetValue == 230 || m_iRetValue == 202))
        {
            Cleanup();
            MessageString = m_sReply;
            throw new IOException(m_sReply.Substring(4));
        }
    }
}
```

## Re: FTP Upload

```
m_bLoggedIn = true;
//call the ChangeDirectory user-defined function to change the folder to
the remote FTP folder that is mapped.
ChangeDirectory(m_sRemotePath);

//return the final result.
return m_bLoggedIn;
}

//if the value of mode is true, set the binary mode for downloads.
Otherwise, set ASCII mode.
public void SetBinaryMode(bool bMode)
{
if(bMode)
{
//send the FTP command to set the binary mode.
//(TYPE is an FTP command that is used to specify representation type.)
SendCommand("TYPE I");
}
else
{
//send the FTP command to set ASCII mode.
//(TYPE is a FTP command that is used to specify representation type.)
SendCommand("TYPE A");
}

if(m_iRetVal != 200)
{
MessageString = m_sReply;
throw new IOException(m_sReply.Substring(4));
}
}

//download a file to the local folder of the assembly, and keep the same
file name.
public void DownloadFile(string sFileName)
{
DownloadFile(sFileName, "", false);
}
//download a remote file to the local folder of the assembly, and keep the
same file name.
public void DownloadFile(string sFileName, bool bResume)
{
DownloadFile(sFileName, "", bResume);
}
//download a remote file to a local file name. You must include a path.
//the local file name will be created or will be overwritten, but the path
must exist.
public void DownloadFile(string sFileName, string sLocalFileName)
{
```

## Re: FTP Upload

```
DownloadFile(sFileName, sLocalFileName, false);
}
//download a remote file to a local file name and include a path. Then,
set the resume flag. The local file name will be created or will be
overwritten, but the path must exist.
public void DownloadFile(string sFileName, string sLocalFileName, bool
bResume)
{
Stream st;
FileStream output;
Socket cSocket;
long offset, npos;

if(!(m_bLoggedIn))
{
Login();
}

SetBinaryMode(true);

if(sLocalFileName.Equals(""))
{
sLocalFileName = sFileName;
}

if(!(File.Exists(sLocalFileName)))
{
st = File.Create(sLocalFileName);
st.Close();
}

output = new FileStream(sLocalFileName, FileMode.Open);
cSocket = CreateDataSocket();
offset = 0;

if(bResume)
{
offset = output.Length;
}

if(offset > 0)
{
//send an FTP command to restart.
SendCommand("REST " + offset);
if(m_iRetVal != 350)
{
offset = 0;
}
}

if(offset > 0)
{
```

## Re: FTP Upload

```
npos = output.Seek(offset, SeekOrigin.Begin);
}
}
//send an FTP command to retrieve a file.
SendCommand("RETR " + sFileName);

if(!(m_iRetVal == 150 || m_iRetVal == 125))
{
    MessageString = m_sReply;
    throw new IOException(m_sReply.Substring(4));
}

while(true)
{
    Array.Clear(m_aBuffer, 0, m_aBuffer.Length);
    m_iBytes = cSocket.Receive(m_aBuffer, m_aBuffer.Length, 0);
    output.Write(m_aBuffer, 0, m_iBytes);

    if(m_iBytes <= 0)
    {
        break;
    }
}

output.Close();
if(cSocket.Connected)
{
    cSocket.Close();
}

ReadReply();
if(!(m_iRetVal == 226 || m_iRetVal == 250))
{
    MessageString = m_sReply;
    throw new IOException(m_sReply.Substring(4));
}
}

//this is a function that is used to upload a file from your local hard
disk to your FTP site.
public void UploadFile(string sFileName)
{
    UploadFile(sFileName, false);
}

//this is a function that is used to upload a file from your local hard
disk to your FTP site and then set the resume flag.
public void UploadFile(string sFileName, bool bResume)
{
    Socket cSocket;
    long offset;
    FileStream input;
```

```
bool bFileNotFound;

if(!(m_bLoggedIn))
{
Login();
}

cSocket = CreateDataSocket();
offset = 0;

if(bResume)
{
try
{
SetBinaryMode(true);
offset = GetFileSize(sFileName);
}
catch(Exception ex)
{
offset = 0;
}
}

if(offset > 0)
{
SendCommand("REST " + offset);
if(m_iRetVal != 350)
{
//the remote server may not support resuming.
offset = 0;
}
}
//send an FTP command to store a file.
SendCommand("STOR " + Path.GetFileName(sFileName));
if(!(m_iRetVal == 125 || m_iRetVal == 150))
{
MessageString = m_sReply;
throw new IOException(m_sReply.Substring(4));
}

//check to see if the file exists before the upload.
bFileNotFound = false;
if(File.Exists(sFileName))
{
//open the input stream to read the source file.
input = new FileStream(sFileName, FileMode.Open);
if(offset != 0)
{
input.Seek(offset, SeekOrigin.Begin);
}
}
```

## Re: FTP Upload

```
//upload the file.
m_iBytes = input.Read(m_aBuffer, 0, m_aBuffer.Length);
while(m_iBytes > 0)
{
cSocket.Send(m_aBuffer, m_iBytes, 0);
m_iBytes = input.Read(m_aBuffer, 0, m_aBuffer.Length);
}
input.Close();
}
else
{
bFileNotFound = true;
}

if(cSocket.Connected)
{
cSocket.Close();
}

//check the return value if the file was not found.
if(bFileNotFound)
{
MessageString = m_sReply;
throw new IOException("The file: " + sFileName + " was not found." + "
Cannot upload the file to the FTP site.");
}

ReadReply();
if(!(m_iRetValue == 226 || m_iRetValue == 250))
{
MessageString = m_sReply;
throw new IOException(m_sReply.Substring(4));
}
}

//delete a file from the remote FTP server.
public bool DeleteFile(string sFileName)
{
bool bResult;

bResult = true;
if(!(m_bLoggedIn))
{
Login();
}
//send an FTP command to delete a file.
SendCommand("DELE " + sFileName);
if(m_iRetValue != 250)
{
bResult = false;
MessageString = m_sReply;
}
```

## Re: FTP Upload

```
}

//return the final result.
return bResult;
}

//rename a file on the remote FTP server.
public bool RenameFile(string sOldFileName, string sNewFileName)
{
    bool bResult;

    bResult = true;
    if(!(m_bLoggedIn))
    {
        Login();
    }
    //send an FTP command to rename a file.
    SendCommand("RNFR " + sOldFileName);
    if(m_iRetValue != 350)
    {
        MessageString = m_sReply;
        throw new IOException(m_sReply.Substring(4));
    }

    //send an FTP command to rename a file to a file name. It will overwrite
    if newFileName exists.
    SendCommand("RNTO " + sNewFileName);
    if(m_iRetValue != 250)
    {
        MessageString = m_sReply;
        throw new IOException(m_sReply.Substring(4));
    }
    //return the final result.
    return bResult;
}

//this is a function that is used to create a folder on the remote FTP
server.
public bool CreateDirectory(string sDirName)
{
    bool bResult;

    bResult = true;
    if(!(m_bLoggedIn))
    {
        Login();
    }
    //send an FTP command to make a folder on the FTP server.
    SendCommand("MKD " + sDirName);
    if(m_iRetValue != 257)
    {
```

## Re: FTP Upload

```
bResult = false;
MessageString = m_sReply;
}

//return the final result.
return bResult;
}

//this is a function that is used to delete a folder on the remote FTP
server.
public bool RemoveDirectory(string sDirName)
{
bool bResult;

bResult = true;
//check if you are logged on to the FTP server.
if(!(m_bLoggedIn))
{
Login();
}
//send an FTP command to remove a folder on the FTP server.
SendCommand("RMD " + sDirName);
if(m_iRetVal != 250)
{
bResult = false;
MessageString = m_sReply;
}

//return the final result.
return bResult;
}

//this is a function that is used to change the current working folder on
the remote FTP server.
public bool ChangeDirectory(string sDirName)
{
bool bResult;

bResult = true;
//check if you are in the root directory.
if(sDirName.Equals("."))
{
return bResult;
}
//check if you are logged on to the FTP server.
if(!(m_bLoggedIn))
{
Login();
}
//send an FTP command to change the folder on the FTP server.
SendCommand("CWD " + sDirName);
```

```
if(m_iRetVal != 250)
{
bResult = false;
MessageString = m_sReply;
}

m_sRemotePath = sDirName;

//return the final result.
return bResult;
}

//close the FTP connection of the remote server.
public void CloseConnection()
{
if(!(m_objClientSocket == null))
{
//send an FTP command to end an FTP server system.
SendCommand("QUIT");
}

Cleanup();
}
#endregion

#region "Private Subs and Functions"
//read the reply from the FTP server.
private void ReadReply()
{
m_sMes = "";
m_sReply = ReadLine();
m_iRetVal = int.Parse(m_sReply.Substring(0, 3));
}

//clean up some variables.
private void Cleanup()
{
if(!(m_objClientSocket == null))
{
m_objClientSocket.Close();
m_objClientSocket = null;
}

m_bLoggedIn = false;
}

//read a line from the FTP server.
private string ReadLine()
{
return ReadLine(false);
}
```

## Re: FTP Upload

```
private string ReadLine(bool bClearMes)
{
    char seperator = '\n';
    string[] mess;

    if(bClearMes)
    {
        m_sMes = "";
    }
    while(true)
    {
        Array.Clear(m_aBuffer, 0, BLOCK_SIZE);
        m_iBytes = m_objClientSocket.Receive(m_aBuffer, m_aBuffer.Length, 0);
        m_sMes += ASCII.GetString(m_aBuffer, 0, m_iBytes);
        if(m_iBytes < m_aBuffer.Length)
        {
            break;
        }
    }

    mess = m_sMes.Split(seperator);
    if(m_sMes.Length > 2)
    {
        m_sMes = mess[mess.Length - 2];
    }
    else
    {
        m_sMes = mess[0];
    }

    if(!(m_sMes.Substring(3, 1).Equals(" ")))
    {
        return ReadLine(true);
    }

    return m_sMes;
}

//this is a function that is used to send a command to the FTP server that
you are connected to.
private void SendCommand(string sCommand)
{
    sCommand = sCommand + '\n';
    byte[] cmdbytes = ASCII.GetBytes(sCommand);
    m_objClientSocket.Send(cmdbytes, cmdbytes.Length, 0);
    ReadReply();
}

//create a data socket.
private Socket CreateDataSocket()
```

## Re: FTP Upload

```
{
int index1, index2, len;
int partCount, i, port;
string ipData, buf, ipAddress;
int[] parts = new int[6];
char ch;
Socket s;
IPEndPoint ep;
//send an FTP command to use a passive data connection.
SendCommand("PASV");
if(m_iRetVal != 227)
{
MessageString = m_sReply;
throw new IOException(m_sReply.Substring(4));
}

index1 = m_sReply.IndexOf("(");
index2 = m_sReply.IndexOf(")");
ipData = m_sReply.Substring(index1 + 1, index2 - index1 - 1);

len = ipData.Length;
partCount = 0;
buf = "";

for(i = 0; i < len && partCount <= 6; i++)
{
ch = char.Parse(ipData.Substring(i, 1));
if(char.IsDigit(ch))
{
buf += ch;
}
else
{
if(ch != ',')
{
MessageString = m_sReply;
throw new IOException("Malformed PASV reply: " + m_sReply);
}
}

if((ch == ',') || (i + 1 == len))
{
try
{
parts[partCount] = int.Parse(buf);
partCount += 1;
buf = "";
}
catch(Exception ex)
{
MessageString = m_sReply;

```

## Re: FTP Upload

```
throw new IOException("Malformed PASV reply: " + m_sReply);
}
}
}

ipAddress = parts[0] + "." + parts[1] + "." + parts[2] + "." + parts[3];

port = parts[4] << 8;

//determine the data port number.
port = port + parts[5];

s = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
ep = new IPEndPoint(Dns.Resolve(ipAddress).AddressList[0], port);

try
{
s.Connect(ep);
}
catch(Exception ex)
{
MessageString = m_sReply;
throw new IOException("Cannot connect to remote server");
//if you cannot connect to the FTP server that is specified, make the
boolean variable false.
flag_bool = false;
}
//if you can connect to the FTP server that is specified, make the
boolean variable true.
flag_bool = true;
return s;
}
#endregion

// #region TestFTP
// private void TestFTP()
// {
// //create an instance of the FTP class that is created.
// clsFTP ff;
//
//
// try
// {
// //pass values to the constructor. These values can be overridden by
// setting the appropriate properties on the instance of the clsFTP class. The
// third parameter is the user name. The FTP site is accessed with the user
// name. If there is no specific user name, the user name can be anonymous. The
// fourth parameter is the password. The FTP server is accessed with the
// password. The fifth parameter is the port of the FTP server. The port of the
// FTP server is typically 21.
//
```

## Re: FTP Upload

```
// ff = new clsFTP(StrIP, "/Myfolder/", "anonymous", "", 21);
//
// //try to log on to the FTP server.
// if(ff.Login() == true)
// {
// //change the directory on your FTP site.
// if(ff.ChangeDirectory("MyOwnFolder") == true)
// {
// //successful changing the directory
// ff.CreateDirectory("FTPFOLDERNEW");
// ff.ChangeDirectory("FTPFOLDERNEW");
//
// ff.SetBinaryMode(true);
//
// //upload a file from your local hard disk to the FTP site.
// ff.UploadFile("C:\Test\Example1.txt");
// ff.UploadFile("C:\Test\Example2.doc");
// ff.UploadFile("C:\Test\Example3.doc");
//
// //download a file from the FTP site to your local hard disk.
// ff.DownloadFile("Example2.doc", "C:\Test\Example2.doc");
//
// //remove a file from the FTP site.
// ff.DeleteFile("Example1.txt");
//
// //rename a file on the FTP site.
// ff.RenameFile("Example3.doc", "Example3_new.doc");
//
// //change the directory to one directory before.
// ff.ChangeDirectory("../");
// }
//
// //create a new directory.
// ff.CreateDirectory("MyOwnFolderNew");
//
// //remove the directory that is created on the FTP site.
// ff.RemoveDirectory("MyOwnFolderNew");
// }
// }
// catch(System.Exception ex)
// {
// }
// finally
// {
// //always close the connection to make sure that there are not any
// not-in-use FTP connections. Check if you are logged on to the FTP server and
// then close the connection.
// if(ff.flag_bool == true)
// {
// ff.CloseConnection();
// }
```

Re: FTP Upload

```
// }  
// }  
// #endregion
```

```
}  
}
```

--

Misbah Arefin

"MDB" wrote:

Anyone have any pointers / links / examples on how to upload files through FTP? I am currently using the built in file upload control however it seems to take a very long time to upload files and would like to explore other ways of doing it. Any pointers or suggestions would greatly be appreciated.