**[Java FTP file download tutorial and example](http://www.codejava.net/java-se/networking/ftp/java-ftp-file-download-tutorial-and-example)**

Last Updated on 15 July 2015   |   [Print](http://www.codejava.net/java-se/networking/ftp/java-ftp-file-download-tutorial-and-example?tmpl=component&print=1&page=) [Email](http://www.codejava.net/component/mailto/?tmpl=component&template=protostar&link=86d3342704a2db545e6262481a52eb4eea155ec2)

[**Java Performance**](http://www.amazon.com/gp/product/0137142528/ref=as_li_tf_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=0137142528&linkCode=as2&tag=code0ac-20)**http://ir-na.amazon-adsystem.com/e/ir?t=code0ac-20&l=as2&o=1&a=0137142528**[**Clean Code**](http://www.amazon.com/gp/product/0132350882/ref=as_li_tf_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=0132350882&linkCode=as2&tag=code0ac-20)**http://ir-na.amazon-adsystem.com/e/ir?t=code0ac-20&l=as2&o=1&a=0132350882**[**The Clean Coder**](http://www.amazon.com/gp/product/0137081073/ref=as_li_tf_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=0137081073&linkCode=as2&tag=code0ac-20)**http://ir-na.amazon-adsystem.com/e/ir?t=code0ac-20&l=as2&o=1&a=0137081073**[**The Pragmatic Programmer**](http://www.amazon.com/gp/product/020161622X/ref=as_li_tf_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=020161622X&linkCode=as2&tag=code0ac-20)**http://ir-na.amazon-adsystem.com/e/ir?t=code0ac-20&l=as2&o=1&a=020161622X**

With the help of [**Apache Commons Net API**](http://commons.apache.org/net), it is easy to write Java code for downloading a file from a remote FTP server to local computer. In this article, you will learn how to properly implement Java code to get files downloaded from a server via FTP protocol. A working sample program also provided.

**Table of Content**

* 1. [Apache Commons Net API for downloading files by FTP protocol](http://www.codejava.net/java-se/networking/ftp/java-ftp-file-download-tutorial-and-example#DownloadAPI)
  2. [The proper steps to download a file](http://www.codejava.net/java-se/networking/ftp/java-ftp-file-download-tutorial-and-example#DownloadSteps)
  3. [Sample program code](http://www.codejava.net/java-se/networking/ftp/java-ftp-file-download-tutorial-and-example#SampleCode)
  4. [Compile and run the sample program](http://www.codejava.net/java-se/networking/ftp/java-ftp-file-download-tutorial-and-example#CompileAndRun)

**1. Apache Commons Net API for downloading files by FTP protocol**

 The org.apache.commons.net.ftp.FTPClient class provides two methods for downloading files from a FTP server:

* + **boolean retrieveFile(String remote, OutputStream local)**: This method retrieves a remote file whose path is specified by the parameter *remote*, and writes it to the OutputStream specified by the parameter *local*. The method returns **true** if operation completed successfully, or **false** otherwise. This method is suitable in case we don’t care how the file is written to disk, just let the system use the given OutputStream to write the file. We should close OutputStream the after the method returns.
  + **InputStream retrieveFileStream(String remote)**: This method does not use an OutputStream, instead it returns an InputStreamwhich we can use to read bytes from the remote file. This method gives us more control on how to read and write the data. But there are two important points when using this method:
    - The method completePendingCommand() must be called afterward to finalize file transfer and check its return value to verify if the download is actually done successfully.
    - We must close the InputStream explicitly.

Which method is used suitable for you? Here are few tips:

* + The first method provides the simplest way for downloading a remote file, as just passing an OutputStream of the file will be written on disk.
  + The second method requires more code to be written, as we have to create a new OutputStream for writing file’s content while reading its byte arrays from the returned InputStream. This method is useful when we want to measure progress of the download, i.e. how many percentages of the file have been transferred. In addition, we have to call the completePendingCommand()to finalize the download.
  + Both the methods throw an IOException exception (or one of its descendants, FTPConnectionClosedExceptionand CopyStreamException). Therefore, make sure to handle these exceptions when calling the methods.

In addition, the following two methods must be invoked before calling the retrieveFile() and retrieveFileStream()methods:

* + void **enterLocalPassiveMode**(): this method switches data connection mode from server-to-client (default mode) to client-to-server which can pass through firewall. There might be some connection issues if this method is not invoked.
  + boolean **setFileType**(int fileType) : this method sets file type to be transferred, either as ASCII text file or binary file. It is recommended to set file type to FTP.BINARY\_FILE\_TYPE, rather than FTP.ASCII\_FILE\_TYPE.

**2. The proper steps to download a file**

 Here are the steps to properly implement code for downloading a remote file from a FTP server using Apache Commons Net API which is discussed so far:

* + [Connect and login to the server](http://www.codejava.net/java-se/networking/ftp/connect-and-login-to-a-ftp-server).
  + Enter local passive mode for data connection.
  + Set file type to be transferred to binary.
  + Construct path of the remote file to be downloaded.
  + Create a new OutputStream for writing the file to disk.
  + If using the first method (retrieveFile):
    - Pass the remote file path and the OutputStream as arguments of the method retrieveFile().
    - Close the OutputStream.
    - Check return value of retrieveFile() to verify success.
  + If using the second method (retrieveFileStream):
    - Retrieve an InputStream returned by the method retrieveFileStream().
    - Repeatedly a byte array from the InputStream and write these bytes into the OutputStream, until theInputStream is empty.
    - Call completePendingCommand() method to complete transaction.
    - Close the opened OutputStream the InputStream.
    - Check return value of completePendingCommand() to verify success.
  + Logout and disconnect from the server.

**3. Sample program code**

In the following sample program, using both methods is implemented for transferring a file from the FTP server to local computer. Here is the program’s source code:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76 | import java.io.BufferedOutputStream;  import java.io.File;  import java.io.FileInputStream;  import java.io.FileOutputStream;  import java.io.IOException;  import java.io.InputStream;  import java.io.OutputStream;    import org.apache.commons.net.ftp.FTP;  import org.apache.commons.net.ftp.FTPClient;    /\*\*   \* A program demonstrates how to upload files from local computer to a remote   \* FTP server using Apache Commons Net API.   \* @author www.codejava.net   \*/  public class FTPDownloadFileDemo {        public static void main(String[] args) {          String server = "www.myserver.com";          int port = 21;          String user = "user";          String pass = "pass";            FTPClient ftpClient = new FTPClient();          try {                ftpClient.connect(server, port);              ftpClient.login(user, pass);              ftpClient.enterLocalPassiveMode();              ftpClient.setFileType(FTP.BINARY\_FILE\_TYPE);                // APPROACH #1: using retrieveFile(String, OutputStream)              String remoteFile1 = "/test/video.mp4";              File downloadFile1 = new File("D:/Downloads/video.mp4");              OutputStream outputStream1 = new BufferedOutputStream(new FileOutputStream(downloadFile1));              boolean success = ftpClient.retrieveFile(remoteFile1, outputStream1);              outputStream1.close();                if (success) {                  System.out.println("File #1 has been downloaded successfully.");              }                // APPROACH #2: using InputStream retrieveFileStream(String)              String remoteFile2 = "/test/song.mp3";              File downloadFile2 = new File("D:/Downloads/song.mp3");              OutputStream outputStream2 = new BufferedOutputStream(new FileOutputStream(downloadFile2));              InputStream inputStream = ftpClient.retrieveFileStream(remoteFile2);              byte[] bytesArray = new byte[4096];              int bytesRead = -1;              while ((bytesRead = inputStream.read(bytesArray)) != -1) {                  outputStream2.write(bytesArray, 0, bytesRead);              }                success = ftpClient.completePendingCommand();              if (success) {                  System.out.println("File #2 has been downloaded successfully.");              }              outputStream2.close();              inputStream.close();            } catch (IOException ex) {              System.out.println("Error: " + ex.getMessage());              ex.printStackTrace();          } finally {              try {                  if (ftpClient.isConnected()) {                      ftpClient.logout();                      ftpClient.disconnect();                  }              } catch (IOException ex) {                  ex.printStackTrace();              }          }      }  } |

**4. Compile and run the sample program**

To compile and run the program, you need an Apache Commons Net’s jar library file presented in the classpath. [Click here](http://commons.apache.org/net/download_net.cgi) to download the latest distribution of Apache Commons Net.

Extract the distribution zip file and copy the commons-net-VERSION.jar file into the same folder of theFTPDownloadFileDemo.java file.

Type the following command to compile the program:

**javac –cp commons-net-VERSION.jar FTPDownloadFileDemo.java**

And type this command to run the program:

**java –cp commons-net-VERSION.jar;. FTPDownloadFileDemo**