عنوان: **نحوه کار با ftp - بخش اول**
نویسنده: بهمن آبادی
تاریخ: ۱۳۹۲/۰۲/۱۱ ۱۹:۵۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, C#, system.net, ftp

امروز می‌خوام نحوه کار با FTP بصورت ساده برای کاربران و برنامه نویسان مبتدی رو آموزش بدم.

برای استفاده از FTP نیاز به یک اکانت FTP در سایت مورد نظر بهمراه دسترسی به پوشه ای مشخص می‌باشد.
برای مثال ما یک اکانت FTP در سایت dotnettip.info داریم که به پوشه upload دسترسی داره.

ابتدا در فایل Web.config و در بین تگ های appSettings مقادیر زیر را برای دسترسی به اکانت و نام کاربری و رمز عبور ذخیره می‌کنیم.

```
<add key="FtpAddress" value="ftp://ftp.dotnetips.info" />
<add key="FtpUser" value="uploadcenter" />
<add key="FtpPass" value="123123" />
<add key="FolderPath" value="~/Upload/" />
```

*نکته : برای امنیت بیشتر و دسترسی به اطلاعات اکانت می‌شود از روش‌های دیگری نیز استفاده کرد.

در ادامه یک کلاس در App_code پروژه خود با نام FTPHelper ایجاد می‌کنیم و کد زیر را در آن قرار می‌دهیم:
تکه کد بالا برای ست کردن مقادیر نام کاربری و رمز عبور و آدرس FTP در کلاس مذکور که بصورت پیش‌فرض از web.config پر می‌شود ایجاد و بکار خواهد رفت.

```
using System.Net;
using System.IO;
using System.Configuration;


public class FtpHelper
{
public FtpHelper()
{
       //Default Value Set From Application
         _hostname = ConfigurationManager.AppSettings.GetValues("FtpAddress")[0];
        _username = ConfigurationManager.AppSettings.GetValues("FtpUser")[0];
        _password = ConfigurationManager.AppSettings.GetValues("FtpPass")[0];
}

    #region "Properties"
    private string _hostname;
    /// <summary>
    /// Hostname
    /// </summary>
    /// <value></value>
    /// <remarks>Hostname can be in either the full URL format
    /// ftp://ftp.myhost.com or just ftp.myhost.com
    /// </remarks>
    public string Hostname
    {
        get
        {
            if (_hostname.StartsWith("ftp://"))
            {
                return _hostname;
            }
            else
            {
                return "ftp://" + _hostname;
            }
        }
        set
        {
            _hostname = value;
        }
    }
    private string _username;
```

```
    /// <summary>
    /// Username property
    /// </summary>
    /// <value></value>
    /// <remarks>Can be left blank, in which case 'anonymous' is returned</remarks>
    public string Username
    {
        get
        {
            return (_username == "" ? "anonymous" : _username);
        }
        set
        {
            _username = value;
        }
    }
    private string _password;
    public string Password
    {
        get
        {
            return _password;
        }
        set
        {
            _password = value;
        }
    }


    #endregion
}
```

سپس فضای نام‌های زیر را در کلاس خود قرار می‌دهیم.

```
using System.Net;
using System.IO;
```

حالا برای بارگذاری فایل می‌توانیم از یک تابع بصورت shared استفاده کنیم که بتوان با دادن آدرس فایل بصورت فیزیکی به تابع و مشخص کردن پوشه مورد نظر آنرا در هاست مقصد (FTP) بارگذاری کرد.توجه داشته باشید که تابع فوق نیازی به قرار گرفتن در کلاس بالا (FtpHelper) ندارد.یعنی می‌توان آنرا در هرجای برنامه پیاده سازی نمود.

```
public static bool Upload(string fileUrl)
        {
            if (File.Exists(fileUrl))
            {
                FtpHelper ftpClient = new FtpHelper();
                string ftpUrl = ftpClient.Hostname + System.IO.Path.GetFileName(fileUrl);


                FtpWebRequest ftp = (FtpWebRequest)FtpWebRequest.Create(ftpUrl);
                ftp.Credentials = new NetworkCredential(ftpClient.Username, ftpClient.Password);

                ftp.KeepAlive = true;
                ftp.UseBinary = true;
                ftp.Timeout = 3600000;
                ftp.KeepAlive = true;
                ftp.Method = WebRequestMethods.Ftp.UploadFile;

                const int bufferLength = 102400;
                byte[] buffer = new byte[bufferLength];
                int readBytes = 0;

                //open file for reading
                using (FileStream fs = File.OpenRead(fileUrl))
                {
                    try
                    {
                        //open request to send
                        using (Stream rs = ftp.GetRequestStream())
```

```
                    {
                        do
                        {
                            readBytes = fs.Read(buffer, 0, bufferLength);
                            fs.Write(buffer, 0, readBytes);
                        } while (!(readBytes < bufferLength));
                        rs.Close();
                    }

                }
                catch (Exception)
                {
                 //Optional Alert for Exeption To Application Layer
                   //throw (new ApplicationException("بارگذاری فایل با خطا  رو به رو شد"));

                }
                finally
                {
                    //ensure file closed
                    //fs.Close();
                }

            }

            ftp = null;
            return true;
        }
        return false;


    }
```

تکه کد بالا فایل مورد نظر را در صورت وجود به صورت تکه‌های 100 کیلوبایتی بر روی ftp بارگذاری می‌کند، که می‌توانید مقدار آنرا نیز تغییر دهید.
اینکار باعث افزایش سرعت بارگذاری در فایل‌های با حجم بالا برای بارگذاری می‌شود.

در بخش‌های بعدی نحوه ایجاد پوشه ، حذف فایل ، حذف پوشه و دانلود فایل از روی FTP را بررسی خواهیم کرد.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۱۱ ۲۲:۳۸

ممنون از مطلب شما.
چند نکته جزئی در مورد کدهای تهیه شده:
- وجود این try/catch در اینجا هیچ هدفی رو برآورده نکرده. از قسمت throw ex هم توصیه می‌شود که استفاده نکنید. از thow خالی استفاده کنید تا stack trace پاک نشه. به علاوه زمانیکه مشغول به طراحی یک کتابخانه هستید تا حد ممکن از ذکر try/catch خودداری کنید. وظیفه بررسی این مسایل مرتبط است به لایه‌های بالاتر استفاده کننده و نه کتابخانه پایه.
- if ابتدای متد هم ضرورتی ندارد. اگر قرار است باشد، باید به استفاده کننده در طی یک استثناء اعلام شود که چرا فایل درخواستی او آپلود نشده. در کل استفاده از متد File.Exists به همراه صدور استثناء در صورت عدم وجود فایل، در اینجا مناسب‌تر است.
- نامگذاری‌هایی مانند obj_ftp مربوط به دوران C است. در سی‌شارپ روش دیگری رو باید استفاده کنید که در مطلب اصول نامگذاری در دات نت به تفصیل بررسی شده.
- بررسی صفر بودن readBytes بهتر است پیش از فراخوانی متد Write انجام شود.
- یک سری از اشیاء در دات نت پیاده سازی کننده IDispoable هستند. به این معنا که بهتر است از using برای استفاده از آن‌ها کمک گرفته شود تا کامپایلر قسمت finally به همراه آزاد سازی منابع را به صورت خودکار اضافه کند. این نکته برای مواردی که در بین کار استثنایی رخ می‌دهد جهت آزاد سازی منابع لازم است. یعنی بهتر بود بجای try/catch از try/finally و یا using در مکان‌های مناسب استفاده می‌شد.
- علت استفاده از شیء Application دراینجا چه چیزی بوده؟ AppSettings خوانده شده از وب کانفیگ برنامه و کل اطلاعات آن در آغاز به کار یک برنامه ASP.NET به صورت خودکار کش می‌شوند. به همین جهت است که اگر حتی یک نقطه در فایل وب کانفیگ تغییر کند برنامه ASP.NET ری استارت می‌شود (تا دوباره تنظیمات را بخواند). بنابراین مستقیما از همان امکانات ConfigurationManager بدون انتساب آن به شیء سراسری Application استفاده کنید. اینکار سربارای آنچنانی هم ندارد؛ چون از حافظه خوانده می‌شود و نه از فایل. هر بار فراخوانی ConfigurationManager.AppSettings به معنای مراجعه به فایل web.config نیست. فقط بار اول اینکار انجام می‌شود؛ تا زمانیکه این فایل تغییر کند یا برنامه ری استارت شود.

نویسنده: بهمن آبادی
تاریخ: ۱۳۹۲/۰۲/۱۲ ۳:۳۰

با تشکر از نظر شما. تمام صحبت‌های شما منطقی است. اولا من این بخش رو می‌خواستم بعدا تکمیل کنم ولی بدلایلی زودتر ارسال کردم.

مواردی هم که اعلام کزدید کاملا صحیح می‌باشد.فقط بخش تابعی که درون آن از try/catch استفاده شده رو میخواستم ابتدا در تیکه کد دیگری از یک صفحه aspx بنویسم که وقتی برای بررسی مجدد تابع پیدا نکردم وگرنه اونجا try/catch بلا استفاده است.

موارد متذکر شده برای سهولت بیشتر کاربران در کدهای فوق لحاظ و ویرایش گردید.
با تشکر.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۲ ۱۱:۴

```
.locals init ([0] class [mscorlib]System.IO.TextWriter w)
  IL_0000:  ldstr      "log.txt"
  IL_0005:  call       class [mscorlib]System.IO.StreamWriter
      [mscorlib]System.IO.File::CreateText(string)
  IL_000a:  stloc.0

 .try
  {
    IL_000b:  ldloc.0
```

```
  IL_000c:  ldstr      "This is line one"
  IL_0011:  callvirt   instance void [mscorlib]
    System.IO.TextWriter::WriteLine(string)
  IL_0016:  leave.s    IL_0022
} // end .try
finally
{
  IL_0018:  ldloc.0
  IL_0019:  brfalse.s  IL_0021
  IL_001b:  ldloc.0
  IL_001c:  callvirt   instance void [mscorlib]
    System.IDisposable::Dispose()
  IL_0021:  endfinally
} // end handler
```

ماخذ

زمانیکه از using statement استفاده می‌کنید، خود کامپایلر try/finally رو اضافه می‌کنه. یعنی قسمت close الان بهش نیازی نیست چون استریم مورد استفاده حتما در اینجا dispose میشه.

نویسنده: بهمن آبادی
تاریخ: ۱۳۹۲/۰۲/۱۲ ۱۲:۵۰

از دستم در رفته ...شما ببخشید.

بیشتر هدفم روند بارگذاری و تکنیک بارگذاری اون بود ولی خوب درست کد نوشتن کاملاً صحیحه و من می‌خواستم برای کاربرای آماتور بنویسم ولی مجبورم کردین که کاربرای حرفه ای هم مثل شما به کد ایرادی نباشه.

بازم ممنون از دقت و راهنماییتون .

حالا بهتر شد نسبت به قبلیه ؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۲ ۱۷:۲۸

من فکر می‌کنم اگر ReSharper رو نصب کنید، پیش از ارائه یک مطلب یا پروژه خیلی از ایرادات رو با خط کشیدن زیر اون یا نمایش یک علامت زرد کنار صفحه گوشزد می‌کنه. مثلا می‌گه که این نوع نامگذاری درست نیست یا این شیء رو میشه با using محصور کرد. خلاصه از دستش ندید، حیفه!

نویسنده: سمیرا قادری
تاریخ: ۱۳۹۲/۰۶/۱۳ ۱۲:۳۷

با تشکر از مطلب خوبتان
ببخشید این روش کار کردن با Ftp چه مزایایی دارد و چه کاربردی دارد؟ چون من تا حالا فقط از Cute Ftp استفاده کردم آن هم برای Upload سایت بعد از Publish

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۶/۱۳ ۱۲:۵۱

یه خورده وقت بذاری، باهاش می‌تونی یک Cute Ftp بنویسی.

نویسنده: رضا منصوری
تاریخ: ۱۳۹۲/۱۲/۲۰ ۱۵:۵۰

با تشکر از مطلب خوبتون ،
برای اینکه به طور مستقیم از file.SaveAs استفاده کنیم و فایل را مستقیم به Ftp آپلود کنیم (نه اینکه از فضای ذخیره شده ای کپی کنیم) چه پیشنهادی می‌دهید ؟ مثلا ما هاست دانلود داریم و میخواهیم فایل‌های ارسالی را در هاست دانلود ذخیره کنیم.