

تا قبل از آمدن html5 امکان آپلود چندین فایل در Asp.net web forms امکان پذیر نبود و کاربران می‌بایستی فایل‌های مورد نظر خود را یکی یکی انتخاب و آپلود می‌کردند که تا حد زیادی سخت و حوصله زیادی هم می‌خواست. اما با معرفی html5 یک attribute به تگ مربوط به آپلود فایل به اسم AllowMultiple افزوده شد که مقادیر قابل قبول این attribute مقادیر بولی true, false می‌باشند. اگر این attribute به تگ‌های مربوط به آپلود فایل اضافه نشود، به صورت پیش فرض قادر خواهید بود که هر بار فقط یک فایل را برای آپلود انتخاب کنید. با مقدار دهی این Attribute با مقدار true این اجازه به شما داده می‌شود که در هر بار بتوانید چندین فایل را به صورت همزمان آپلود نمایید.

کد زیر نمایشی از چگونگی بکاربردن این attribute در تگ input می‌باشد:

```
<input type="file" multiple="multiple" name="FileUpload1" id="FileUpload1" />
```

این Attribute از نسخه ASP.NET 4.5 Framework به Asp.net Web forms اضافه شده و برنامه نویسان می‌توانند در صورت نیاز از این امکان استفاده کنند. کد زیر، نحوه افزودن این Attribute به تگ FileUpload در ASP.Net Web Forms می‌باشد:

```
<asp:FileUpload runat="server" ID="FileUploadMultiple" AllowMultiple="true" />
```

و اما به چه روشی می‌توانیم فایل یا فایل‌های انتخاب شده را در این حالت آپلود کرد؟ در صورتیکه از این روش در پروژه‌هایتان استفاده کنید فقط کافیت با یک حلقه تمامی کنترل‌های مورد نظر را پیمایش و هر کدام از فایل‌ها را آپلود و ذخیره نمایید.

برای درک بهتر مطلب، پروژه جدیدی در Asp.net web Forms ایجاد کرده و کنترل‌های زیر را به آن اضافه کنید:

```
<asp:FileUpload runat="server" ID="FileUploadMultiple" AllowMultiple="true" />
<asp:Button runat="server" ID="btnUlpad" Text="Upload" OnClick="btnUlpad_Click" />
<asp:Label runat="server" ID="lblMessage"></asp:Label>
```

و در رویداد مربوط به دکمه آپلود، قطعه کد زیر را قرار دهید :

```
int Count = 0;
foreach (var item in FileUploadMultiple.PostedFiles)
{
    string Extension = Path.GetExtension(item.FileName);
    string FileName = new Random().Next(1, 50).ToString()+Extension;
    item.SaveAs(Server.MapPath("~/")+"//File//"+FileName);
    Count++;
}
if (Count == FileUploadMultiple.PostedFiles.Count)
    lblMessage.Text = string.Format("فایل‌های انتخابی با موفقیت آپلود شدند");
else
    lblMessage.Text = string.Format("{1} از {0} فایل با موفقیت آپلود شد", Count, FileUploadMultiple.PostedFiles.Count);
```

کدهای پروژه : [Upload-multiple.rar](#)

نظرات خوانندگان

نویسنده: علی قربانی
تاریخ: ۱۸:۱۷ ۱۳۹۴/۰۴/۰۷

با سلام؛ کد شما فقط برای ذخیره فایل است اگر بخواهیم آدرس و اسم فایل رو در دیتابیس ذخیره کنیم به چه صورت باید عمل کنیم؟

نویسنده: وحید نصیری
تاریخ: ۱۹:۱۴ ۱۳۹۴/۰۴/۰۷

آدرس نهایی ذخیره شده، همان پارامتر متد SaveAs است. نام هم از item.FileName استخراج می‌شود. این دو مقدار را دارید، مانند روش‌های متداول، ذخیره‌اش کنید.

نویسنده: سید محمد حسین موسوی
تاریخ: ۱۴:۲۲ ۱۳۹۴/۰۴/۰۸

سلام
راهی وجود داره که بشه ظاهر فایل آپلود رو هم سفارشی کرد؟

نویسنده: علی یگانه مقدم
تاریخ: ۱۰:۲۰ ۱۳۹۴/۰۴/۰۹

[استایل و +](#)

نویسنده: عثمان رحیمی
تاریخ: ۱۴:۱۵ ۱۳۹۴/۰۴/۱۰

راه حل مورد نظر برای این کار گذاشتن یک عکس و یا هر تگ دیگر است و در رویداد کلیک تگ مورد نظر، کلیک FileUpload رو فراخوانی کنید: [^](#)

نویسنده: سید محمد حسین موسوی
تاریخ: ۱۶:۱۵ ۱۳۹۴/۰۴/۱۷

اگر قرار باشه همین کار و با fileupload خود html انجام بدیم، موقع ذخیره کردن چی باید بنویسیم؟
چون وقتی می‌نویسیم btnFileUpload.PostedFiles خطا می‌گیره

System.Web.UI.HtmlControls.HtmlInputFile' does not contain a definition for 'PostedFiles' and no extension method 'PostedFiles' accepting a first argument of type 'System.Web.UI.HtmlControls.HtmlInputFile' could be found (are you missing a using directive or an assembly reference?)

نویسنده: عثمان رحیمی
تاریخ: ۲۰:۷ ۱۳۹۴/۰۴/۱۷

ابتدا تگ input خود را در فرم قرار دهید :

```
<input type="file" multiple="multiple" name="File1" id="File1" runat="server" />
```

و در پشت فرم می‌توانید با استفاده از یک حلقه، همه فایل‌های انتخاب شده را آپلود کنید :

```
for (int i = 1; i <= Request.Files.Count; i++)
{
```

```
        var file = Request.Files[i];  
    }
```

برای یکی از پروژه‌ها نیاز به یک آپلودر داشتم که قابلیت Drag&Drop را نیز داشته باشد و در ضمن پیاده سازی آسانی هم داشته باشد. در این بین به تعدادی از کتابخانه‌های جی کوئری می‌پردازیم.

FileDrop

اولین کتابخانه‌ای که با آن آشنا شدم و از آن استفاده کردم، کتابخانه‌ی FileDrop است که بسیار ساده و در عین حال قابلیت‌های خوبی را می‌دهد و از فناوری [FileReader](#) (+) در Htm15 برای اینکار استفاده می‌کند. مرورگرهای کروم، فایرفاکس 3.6 به بعد، IE10 به بعد و Opera 12 به بعد از آن پشتیبانی می‌کنند.

فایل‌های مورد نیاز را از [اینجا دانلود کنید](#) . فایل اسکریپت آن را ابتدا صدا بزنید:

```
<script src="~/scripts/jquery.filedrop.js" type="text/javascript"></script>
```

سپس المان‌های زیر را به کد HTML خود اضافه کنید:

```
<div id="dropZone">بکشانید</div>
<br>
فایل یا فایل‌های آپلود شده:
<ul id="uploadResult"></ul>
```

تگ اول، محلی است که فایل‌ها به سمت آن درگ و روی آن دراپ می‌شوند که از این به بعد به آن محل آپلود می‌گوییم. المان بعدی جهت گزارش فایل‌هایی است که آپلود شده‌اند. با آپلود شدن هر تعداد فایل، اسم آن به لیست اضافه می‌گردد. کدهای css زیر را هم به صفحه اضافه کنید تا محل آپلود زیباتر شود:

```
.files {
    min-height: 42px;
    background: #CCC none repeat scroll 0% 0%;
    border-top: 1px solid #FFF;
    margin: 11px 0px;
    padding: 11px 13px;
    border-radius: 6px;
}

#dropZone.mouse-over {
    background-color: #1d4257;
}
```

کد جی کوئری زیر را به صفحه اضافه کنید:

```
$('#dropZone').filedrop({
    url: uploadAddress,
    paramname: 'files',
    maxFiles: 1,
    dragOver: function() {
        $('#dropZone').addClass('mouse-over');
    },
    dragLeave: function() {
        $('#dropZone').removeClass('mouse-over');
    },
    drop: function() {
        $('#dropZone').removeClass('mouse-over');
    },
    afterAll: function() {
        $('#dropZone').html('با موفقیت انجام شد');
    }
});
```

```

    },
    uploadFinished: function(i, file, response, time) {
        $('#uploadResult').append('<li>' + file.name + '</li>');
    }
});

```

ابتدا پلاگین جی کوئری را روی تگ مربوطه اعمال می‌کنیم و سپس پارامترها را با مشخصات زیر اعمال می‌کنیم:

Url	آدرسی که قرار است فایل‌ها به آن سمت ارسال شوند.
Paramname	در سمت سرور باید فایل‌ها را با استفاده از این نام پارامتر دریافت کنید.
maxFiles	تعداد فایل‌هایی که میتوان با درگ و دراپ کردن روی آن به دست آورد. در بالا به یک فایل محدود شده است.
dragOver	این رویداد زمانی اجرا خواهد شد که اشاره گر با حالت درگ کرده فایل‌ها را به محل آپلود آورده است.
dragLeave	موقعی که ماوس از محل آپلود خارج می‌شود
drop	موقعی که شما فایل‌ها را روی محل آپلود رها می‌کنید.
afterAll	بعد از اینکه همه کارها تمام شد اجرا می‌شود. (آخرین رویداد)
uploadFinished	کار آپلود به پایان رسیده است. در مثال بالا پس از پایان آپلود، نام فایل آپلود شده را به کاربر نشان داده‌ایم.

نحوه‌ی دریافت آن در سمت سرور، در یک اکشن متد به صورت زیر است:

```

[HttpPost]
public virtual ActionResult UpdateApp(IEnumerable<HttpPostedFileBase>files)
{
    foreach (HttpPostedFileBase file in files)
    {
        string filePath = Path.Combine(TempPath, file.FileName);
        file.SaveAs(filePath);
    }

    return Json(new {state = "success", message = "با موفقیت عملیات ارسال فایل انجام شد"},
        JsonRequestBehavior.AllowGet);
}

```

در اکشن متد بالا ما فایل‌ها را از طریق نام پارامتر files که مشخص کرده بودیم، به عنوان یک لیست شمارشی دریافت می‌کنیم. کدها بالا برای ساده‌ترین راه اندازی ممکن کفایت می‌کنند.

این موارد از اصلی‌ترین‌ها هستند که به کار می‌آیند. به غیر این‌ها یک سری خصوصیات اضافه‌تری هم برای آن وجود دارد.

اگر دوست دارید این آپلودر را نیز به یک آپلودر معمولی اتصال دهید از این شناسه استفاده کنید.	fallback_id
با استفاده از کوکی‌ها یک درخواست cross-origin ایجاد می‌کند.	withCredentials
اگر دوست دارید به همراه فایل‌ها اطلاعات دیگری هم به همراه آن ارسال و پست شوند از این طریق اقدام نمایید. می‌تواند در قالب یک متغیر باشد یا خروجی یک تابع.	data

اگر دوست دارید این آپلودر را نیز به یک آپلودر معمولی اتصال دهید از این شناسه استفاده کنید.	fallback_id
<pre>data: { param1: 'value1', param2: function(){ return calculated_data; } }</pre>	
برای ارسال مقدار اضافه‌تر در هدر درخواست به کار میرود و صدا زدن آن همانند کد data می‌باشد.	headers
<p>در صورتیکه در فرایند آپلود خطایی رخ دهد، اجرا می‌گردد. نحوه‌ی کدنویسی آن و بررسی خطاهای آن به شرح زیر است:</p> <pre>error: function(err, file) { switch(err) { case 'BrowserNotSupported': alert('پشتیبانی این فناوری مرورگر از این فناوری پشتیبانی نمی‌کند'); break; case 'TooManyFiles': // قصد آپلود همزمان فایل‌های // بیشتری از حد مجاز تعیین شده دارید break; case 'FileTooLarge': // حداقل حجم یکی از فایل‌ها از حجم // مجاز تعیین شده بیشتر است // برای دسترسی به نام آن فایل از کد // زیر استفاده کنید //file.name break; case 'FileTypeNotAllowed': // نوع حداقل یکی از فایل‌ها با نوع‌ها // مشخص شده ما یکی نیست break; case 'FileExtensionNotAllowed': // پسوند حداقل یکی از فایل‌ها مورد // تایید نیست break; default: break; } }</pre>	error
نوع فایل‌های مجاز را تعیین می‌کند:	
<pre>allowedfiletypes: ['image/jpeg', 'image/png', 'image/gif']</pre>	allowedfiletypes
پسوند فایل‌هایی که برای آپلود مجاز هستند را معرفی می‌کند.	
<pre>allowedfileextensions: ['.apk', '.jar']</pre>	allowedfileextensions
حداکثر حجم مجاز برای هر فایل که به مگابایت بیان می‌شود.	maxfilesize
این رویداد زمانی اجرا می‌شود که فایل‌های درگ شده شما وارد محیط یا پنجره مرورگر می‌شود.	docOver
این رویداد زمانی اجرا می‌گردد که فرایند آپلود هر فایل به طور جداگانه در حال آغاز شدن است:	
متغیر i در کد زیر شامل اندیس فایلی است که آپلودش آغاز شده است و این اندیس از صفر آغاز می‌شود.	
متغیر file دسترسی شما را به اطلاعات یک فایل باز میکند	uploadStarted

<p>اگر دوست دارید این آپلودر را نیز به یک آپلودر معمولی اتصال دهید از این شناسه استفاده کنید.</p>	<p>fallback_id</p>
<p>مانند نام فایل.</p> <p>متغیر len تعداد فایل هایی را که کاربر در محل آپلود رها کرده است، باز میگرداند.</p> <pre>function(i, file, len){ },</pre>	
<p>با اتمام آپلود هر فایل، این رویداد فراخوانی می‌گردد. دو پارامتر اول آن، همانند سابق هستند. پارامتر response خروجی json ایی را که در سمت سرور برگرداندیم، به ما باز می‌گرداند. پارامتر بعدی، زمانی را که برای آپلود طول کشیده است، بر می‌گرداند.</p> <pre>function(i, file, response, time) { }</pre>	<p>uploadFinished</p>
<p>این رویداد برای نمایش پیشرفت یک آپلود مناسب است که آخرین پارامتر آن یک عدد صحیح از پیشرفت فایل را بر می‌گرداند.</p> <pre>function(i, file, progress) { },</pre>	<p>progressUpdated</p>
<p>این رویداد میزان پیشرفت کلیه فایل‌ها را به درصد باز می‌گرداند:</p> <pre>function(progress) { \$('#progress div') .width(progress+"%"); }</pre>	<p>globalProgressUpdated</p>
<p>سرعت آپلود هر فایل را با کیلوبیت بر ثانیه مشخص می‌کند.</p> <pre>function(i, file, speed) { }</pre>	<p>speedUpdated</p>
<p>در صورتی که قصد تغییر نام فایل ارسالی را دارید می‌توانید از این رویداد استفاده کنید. پارامتر name، نام اصلی فایل را بر می‌گرداند که می‌توانید آن را دستکاری کنید و نام جدیدی را به عنوان خروجی برگردانید. نمونه کاربردی از این رویداد</p> <pre>rename: function(name) { }</pre>	<p>rename</p>
<p>این رویداد قبل از آپلود هر فایل آغاز می‌گردد و برگرداندن مقدار false در آن باعث جلوگیری و کنسل شدن آپلود آن فایل می‌گردد.</p> <pre>function(file) { }</pre>	<p>beforeEach</p>

اگر دوست دارید این آپلودر را نیز به یک آپلودر معمولی اتصال دهید از این شناسه استفاده کنید.	fallback_id
پارامترهای اولی تکراری هستند ولی آخرین پارامتر یک تابع done را می‌توان به آن پاس کرد که قبل از اجرای کل عملیات آپلود صدا زده می‌شود.	beforeSend
<pre>function(file, i, done) { }</pre>	

رویدادی به اسم queuefiles هم هست تعداد فایل‌هایی را که می‌توانند به طور موازی و همزمان آپلود گردند، مشخص می‌کند. ولی در این حالت maxfiles مورد استفاده قرار نمی‌گیرد. جهت بررسی یک مثال عملی و همچنین کدهای سمت سرور در PHP می‌توانید از این [آموزش](#) استفاده کنید. با تستی که به صورت لوکال رو آن انجام دادم به نظر نمی‌رسد برای فایل‌های با حجم متوسط به بالا مناسب باشد و برای فایل‌های با حجم کم مناسب می‌باشد. یک فایل 8 مگابایتی در حالت لوکال 9 ثانیه آپلود آن زمان برد و برای فایل‌های بزرگتر، فایرفاکس دیالوگ Stop Script را نشان داد.

PlUpload

این [کتابخانه متن باز](#) هم بسیار کارآمد و ساده و قابل انعطاف است و [مثالهای آماده](#) زیادی دارد. سایت [سابسن](#) هم در بخش آپلود زیرنویس‌ها از این [کتابخانه](#) استفاده می‌کند. از آنجا که آموزش این کتابخانه در [سایت جاری](#) آمده است از ذکر نکات بیشتر در مورد آن خودداری می‌نمایم.

Bootstrap FileStyle

اگر از قالب بوت استراپ استفاده می‌کنید و دوست دارید روی المان input file قدیمی، ولی به شکلی مدرن کار کنید این [کتابخانه](#) هم فراموش نشود.

DropZoneJS

این کتابخانه به نسبت DropFile امکانات بیشتری را دارد و در [سایت اختصاصی](#) آن مثال‌ها و مستندات خوبی قرار گرفته است. در ساده‌ترین حالت آن ابتدا [فایل کتابخانه](#) را صدا زده و سپس تگ فرم را به آن نسبت دهید:

```
<script src="https://rawgit.com/enyo/dropzone/master/dist/dropzone.js"></script>
<form action="/upload-target" class="dropzone"></form>
```

ولی اگر بخواهید آن را به سمت سرور ارسال کنید و از آنجا آن را کنترل کنید، کد فرم را به شکل زیر تغییر دهید:

ابتدا بسته‌ی نیوگت آن را صدا بزنید:

```
Install-Package dropzone
```

با نصب این کتابخانه یک سری فایل CSS هم به سیستم اضافه می‌شود که می‌توانید برای استایل دهی هر چه بیشتر از آن بهره ببرید. کد فرم را به شکل زیر تغییر دهید:

```
<form action="~/Home/SaveUploadedFile" method="post" enctype="multipart/form-data" class="dropzone"
id="dropzoneForm" style="width: 50px; background: none; border: none;">
  <div class="fallback">
    <input name="file" type="file" multiple />
    <input type="submit" value="Upload" />
  </div>
</form>
```

تگی که با کلاس fallback مزین شده است موقعی به کار می‌آید که مرورگر از این کتابخانه پشتیبانی نکرده و مجبور به نمایش یک آپلود معمولی می‌شویم.

با استفاده از کدنویسی هم می‌توان یک المان را به یک آپلودر تبدیل کرد:


```
var myDropzone = new Dropzone("div#myId", { url: "/file/post"});  
//===== OR =====  
$("#div#myId").dropzone({ url: "/file/post" });
```

همانطور که می‌بینید الزامی برای اینکه از یک تگ فرم استفاده کنید ندارد.
برای کانفیگ آپلودرهایی که از طریف المانهای HTML ایجاد می‌شوند، می‌توان از کد زیر استفاده کرد و یک تنظیم عمومی برای تمامی آپلودرهای HTML آن صفحه ایجاد کرد.

```
Dropzone.options.myId= {  
  paramName: "file", //انتقال می‌بند  
  maxFileSize: 2, // MB  
  accept: function(file, done) {  
    if (file.name == "justinbieber.jpg") {  
      done("Naha, you don't.");  
    }  
    else { done(); }  
  }  
};
```

تابع بالا یک آرگومان از نوع file را برگردانده و اگر این تابع، تابع done را با پارامتری رشته‌ای صدا بزند، عملیات آپلود آن فایل کنسل شده و پیام خطایی را نمایش می‌دهد و در صورتیکه بدون پارامتر صدا زده شود، عمل آپلود بدون مشکل انجام می‌شود.
از آنجا که این کتابخانه از تنظیمات وسیعی استفاده می‌کند و از حوصله‌ی این مقاله خارج است، بهتر هست که صفحه‌ی مستندات آن را که کامل هم هست، مطالعه بفرمایید. از سری قابلیت‌هایی که پشتیبانی می‌کند: موارد پوشش داده شده در FileDrop، ساخت layout، ایجاد صف، متد حذف و اضافه و از این قبیل، ایجاد تصویر تمبر مانند و ...
یک نکته تکمیلی در مورد آپلود: در ASP.net به طور پیش فرض نهایت حجم فایل آپلودی 4 مگابایتی تعیین شده است که می‌توانید آن را از طریق web.config تغییر دهید:

```
<configuration>  
  <system.web>  
    <httpRuntime maxRequestLength="1048576" />  
  </system.web>  
</configuration>
```

برای IIS 7 به بعد هم از تکه کد زیر استفاده کنید:

```
<system.webServer>  
  <security>  
    <requestFiltering>  
      <requestLimits maxAllowedContentLength="1073741824" />  
    </requestFiltering>  
  </security>  
</system.webServer>
```

در هر دو کد بالا نهایت حجم بر روی یک گیگابایت تعیین شده است که `maxRequestLength` به صورت کیلوبایت و `maxAllowContentLength` به صورت بایت تعیین شده است. توصیه می‌شود هر دو شکل آن را وارد کنید. به خصوص که IIS Express از کد ابتدایی استفاده می‌کند و بخواهید نتیجه‌ی آن را در تست‌ها ببینید.

نظرات خوانندگان

نویسنده:

غلامرضا ربال

تاریخ:

۱۶:۴۱ ۱۳۹۴/۰۴/۳۰

[این کتابخانه](#) هم برای drag&drop جالب است. قابلیت ارسال اجکسی به صورت توکار ساپورت میشود.