عنوان: **بررسی ویجت Kendo UI File Upload**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۰۳ ۱۱:۳۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Kendo UI

Kendo UI به همراه یک ویجت وب مخصوص ارسال فایل‌ها به سرور نیز هست. این ویجت قابلیت ارسال چندین فایل با هم را به صورت Ajax ایی دارا است و همچنین کاربران می‌توانند فایل‌ها را با کشیدن و رها کردن بر روی آن، به لیست فایل‌های قابل ارسال اضافه کنند.
ارسال فایل Ajax ایی آن توسط HTML5 File API صورت می‌گیرد که در تمام مرورگرهای جدید پشتیبانی خوبی از آن وجود دارد. در مرورگرهای قدیمی‌تر، به صورت خودکار همان حالت متداول ارسال همزمان فایل‌ها را فعال می‌کند (یا همان post back معمولی).

فعال سازی مقدماتی kendoUpload

ابتدایی‌ترین حالت کار با kendoUpload، فعال سازی حالت post back معمولی است؛ به شرح زیر:

```
<form method="post" action="submit" enctype="multipart/form-data">
  <div>
    <input name="files" id="files" type="file" />
    <input type="submit" value="Submit" class="k-button" />
  </div>
</form>
<script>
  $(document).ready(function() {
     $("#files").kendoUpload();
  });
</script>
```

در این حالت صرفا input با نوع file، با ظاهری سازگار با سایر کنترل‌های Kendo UI به نظر می‌رسد و عملیات ارسال فایل، همانند قبل به همراه یک post back است. این روش برای حالتی مفید است که بخواهید یک فایل را به همراه سایر عناصر فرم در طی یک مرحله به سمت سرور ارسال کنید.

فعال سازی حالت ارسال فایل Ajax ایی kendoUpload

برای فعال سازی ارسال Ajax ایی فایل‌ها در Kendo UI نیاز است خاصیت async آن‌را به نحو ذیل مقدار دهی کرد:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
                name: "files",
                async: { // async configuration
                    saveUrl: "@Url.Action("Save", "Home")", // the url to save a file is '/save'
                    removeUrl: "@Url.Action("Remove", "Home")", // the url to remove a file is
'/remove'
                    autoUpload: false, // automatically upload files once selected
                    removeVerb: 'POST'
                },
                multiple: true,
                showFileList: true
            });
        });
    </script>
```

در اینجا دو آدرس ذخیره سازی فایل‌ها و همچنین حذف آن‌ها را مشاهده می‌کنید. امضای این دو اکشن متد در ASP.NET MVC به صورت ذیل هستند:

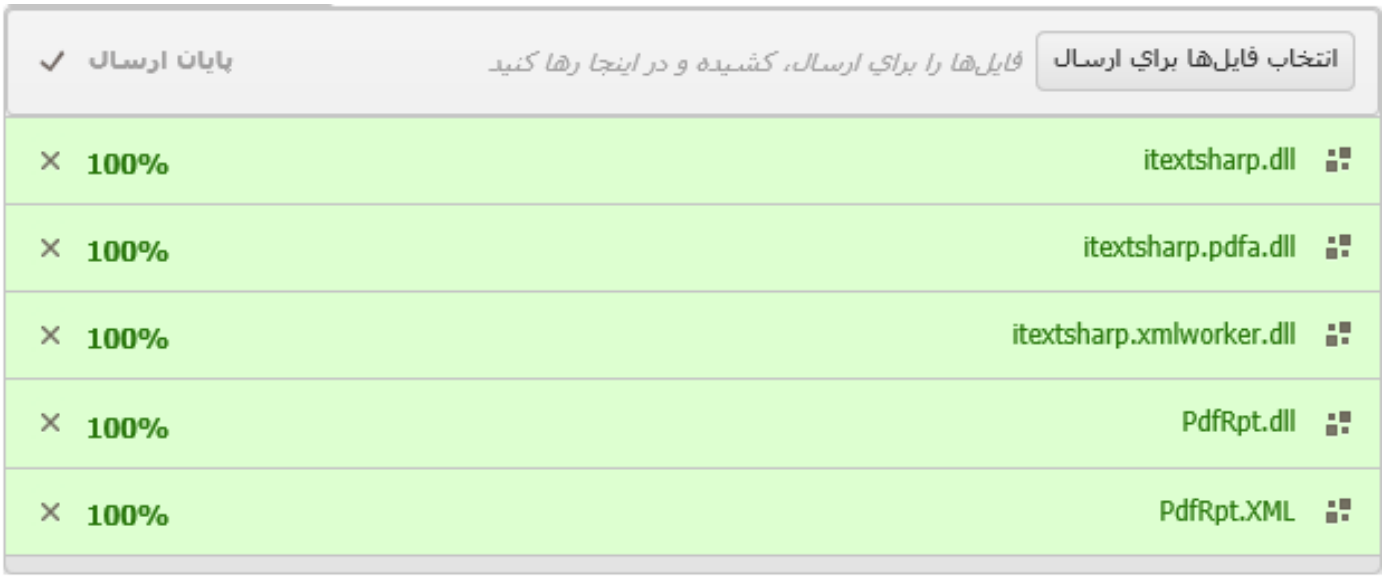
```
[HttpPost]
        public ActionResult Save(IEnumerable<HttpPostedFileBase> files)
        {
```

```
            if (files != null)
            {
                // ...
                // Process the files and save them
                // ...
            }

            // Return an empty string to signify success
            return Content("");
        }

        [HttpPost]
        public ContentResult Remove(string[] fileNames)
        {
            if (fileNames != null)
            {
                foreach (var fullName in fileNames)
                {
                    // ...
                    // delete the files
                    // ...
                }
            }

            // Return an empty string to signify success
            return Content("");
        }
```

در هر دو حالت، لیستی از فایل‌ها توسط kendoUpload به سمت سرور ارسال می‌شوند. در حالت Save، محتوای این فایل‌ها جهت ذخیره سازی بر روی سرور در دسترس خواهد بود. در حالت Remove، صرفا نام این فایل‌ها برای حذف از سرور، توسط کاربر ارسال می‌شوند.
دو دکمه‌ی حذف با کارکردهای متفاوت در ویجت kendoUpload وجود دارند. در ابتدای کار، پیش از ارسال فایل‌ها به سرور:

کلیک بر روی دکمه‌ی حذف در این حالت، صرفا فایلی را از لیست سمت کاربر حذف می‌کند.

پس از ارسال فایل‌ها به سرور:



اما پس از پایان عملیات ارسال، اگر کاربر بر روی دکمه‌ی حذف کلیک کند، توسط آدرس مشخص شده توسط خاصیت removeUrl، نام فایل‌های مورد نظر، برای حذف از سرور ارسال می‌شوند.

چند نکته‌ی تکمیلی

- تنظیم خاصیت autoUpload به true سبب می‌شود تا پس از انتخاب فایل‌ها توسط کاربر، بلافاصله و به صورت خودکار عملیات ارسال فایل‌ها به سرور آغاز شوند. اگر به false تنظیم شود، دکمه‌ی ارسال فایل‌ها در پایین لیست نمایش داده خواهد شد.
- شاید علاقمند باشید تا removeVerb را به DELETE تغییر دهید؛ بجای POST. به همین منظور می‌توان خاصیت removeVerb در اینجا مقدار دهی کرد.
- با تنظیم خاصیت multiple به true، کاربر قادر خواهد شد تا توسط صفحه‌ی دیالوگ انتخاب فایل‌ها، قابلیت انتخاب بیش از یک فایل را داشته باشد.
- showFileList نمایش لیست فایل‌ها را سبب می‌شود.

تعیین پسوند فایل‌های صفحه‌ی انتخاب فایل‌ها

هنگامیکه کاربر بر روی دکمه‌ی انتخاب فایل‌ها برای ارسال کلیک می‌کند، در صفحه‌ی دیالوگ باز شده می‌توان پسوندهای پیش فرض مجاز را نیز تعیین کرد.
برای این منظور تنها کافی است ویژگی accept را به input از نوع فایل اضافه کرد. چند مثال در این مورد:

```
<!-- Content Type with wildcard.  All Images -->
<input type="file" id="demoFile" title="Select file" accept="image/*" />

<!-- List of file extensions -->
<input type="file" id="demoFile" title="Select file" accept=".jpg,.png,.gif" />

<!-- Any combination of the above -->
<input type="file" id="demoFile" title="Select file" accept="audio/*,application/pdf,.png" />
```

نمایش متن کشیدن و رها کردن، بومی سازی برچسب‌ها و نمایش راست به چپ

همانطور که در تصاویر فوق ملاحظه می‌کنید، نمایش این ویجت راست به چپ و پیام‌های آن نیز ترجمه شده‌اند.
برای راست به چپ سازی آن مانند قبل تنها کافی است input مرتبط، در یک div با کلاس k-rtl محصور شود:

```
<div class="k-rtl k-header">
            <input name="files" id="files" type="file"  />
        </div>
```

برای بومی سازی پیام‌های آن می‌توان مانند مثال ذیل، خاصیت localization را مقدار دهی کرد:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
                name: "files",
                async: {
                 //...
                },
                //...
                localization: {
                    select: 'انتخاب فایل‌ها برای ارسال',
                    remove: 'حذف فایل',
                    retry: 'سعی مجدد',
                    headerStatusUploading: 'در حال ارسال فایل‌ها',
                    headerStatusUploaded: 'پایان ارسال',
                    cancel: "لغو",
                    uploadSelectedFiles: "ارسال فایل‌ها",
                    dropFilesHere: "فایل‌ها را برای ارسال، کشیده و در اینجا رها کنید",
                    statusUploading: "در حال ارسال",
                    statusUploaded: "ارسال شد",
                    statusWarning: "اخطار",
                    statusFailed: "خطا در ارسال"
                }
            });
        });
    </script>
```

به علاوه متن dropFilesHere به صورت پیش فرض نامرئی است. برای نمایش آن نیاز است CSS موجود را بازنویسی کرد تا em مرتبط مرئی شود:

```
<style type="text/css">
div.k-dropzone {
    border: 1px solid #c5c5c5; /* For Default; Different for each theme */
}

div.k-dropzone em {
    visibility: visible;
}
</style>
```

تغییر قالب نمایش لیست فایل‌ها

لیست فایل‌ها در ویجت kendoUpload دارای یک قالب پیش فرض است که امکان بازنویسی کامل آن وجود دارد. ابتدا نیاز است یک kendo-template را بر این منظور تدارک دید:

```
<script id="fileListTemplate" type="text/x-kendo-template">
        <li class='k-file'>
            <span class='k-progress'></span>
            <span class='k-icon'></span>
            <span class='k-filename' title='#=name#'>#=name# (#=size# bytes)</span>
            <strong class='k-upload-status'></strong>
        </li>
    </script>
```

و سپس برای استفاده از آن خواهیم داشت:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
```

```
                name: "files",
                async: {
                // ...
                },
                // ...
                template: kendo.template($('#fileListTemplate').html()),
                // ...
            });
        });
    </script>
```

در این قالب، مقدار size هر فایل نیز در کنار نام آن نمایش داده می‌شود.

رخدادهای ارسال فایل‌ها

افزونه‌ی kendoUpload در حالت ارسال Ajax ایی فایل‌ها، رخدادهایی مانند شروع به ارسال، موفقیت، پایان، درصد ارسال فایل‌ها و امثال آن‌را نیز به همراه دارد که لیست کامل آن‌ها را در ذیل مشاهده می‌کنید:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
                name: "files",
                async: { // async configuration
                //...
                },
                //...
                localization: {
                },
                cancel: function () {
                    console.log('Cancel Event.');
                },
                complete: function () {
                    console.log('Complete Event.');
                },
                error: function () {
                    console.log('Error uploading file.');
                },
                progress: function (e) {
                    console.log('Uploading file ' + e.percentComplete);
                },
                remove: function () {
                    console.log('File removed.');
                },
                select: function () {
                    console.log('File selected.');
                },
                success: function () {
                    console.log('Upload successful.');
                },
                upload: function (e) {
                    console.log('Upload started.');
                }
            });
        });
    </script>
```

ارسال متادیتای اضافی به همراه فایل‌های ارسالی

فرض کنید می‌خواهید به همراه فایل‌های ارسالی به سرور، پارامتر codeId را نیز ارسال کنید. برای این منظور باید خاصیت e.data رویداد upload را به نحو ذیل مقدار دهی کرد:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
                name: "files",
                async: {
                //...
```

```
                },
                //...
                localization: {
                },
                upload: function (e) {
                    console.log('Upload started.');
                    // Sending metadata to the save action
                    e.data = {
                        codeId: "1234567",
                        param2: 12
                        //, ...
                    };
                }
            });
        });
    </script>
```

سپس در سمت سرور، امضای متد Save بر اساس پارامترهای تعریف شده در سمت کاربر، به نحو ذیل تغییر می‌کند:

```
[HttpPost]
  public ActionResult Save(IEnumerable<HttpPostedFileBase> files, string codeId)
```

فعال سازی ارسال batch

اگر در متد Save سمت سرور یک break point قرار دهید، مشاهده خواهید کرد که به ازای هر فایل موجود در لیست در سمت کاربر، یکبار متد Save فراخوانی می‌شود و عملا متد Save، لیستی از فایل‌ها را در طی یک فراخوانی دریافت نمی‌کند. برای فعال سازی این قابلیت تنها کافی است خاصیت batch را به true تنظیم کنیم:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
                name: "files",
                async: {
                    // ....
                    batch: true
                },
            });
        });
    </script>
```

به این ترتیب دیگر لیست فایل‌ها به صورت مجزا در سمت کاربر نمایش داده نمی‌شود و تمام آن‌ها با یک کاما از هم جدا خواهند شد. همچنین دیگر شاهد نمایش درصد پیشرفت تکی فایل‌ها نیز نخواهیم بود و اینبار درصد پیشرفت کل batch گزارش می‌شود.
در یک چنین حالتی باید دقت داشت که تنظیم maxRequestLength در web.config برنامه الزامی است؛ زیرا به صورت پیش فرض محدودیت 4 مگابایتی ارسال فایل‌ها توسط ASP.NET اعمال می‌شود:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.web>
    <!-- The request length is in kilobytes, execution timeout is in seconds  -->
    <httpRuntime maxRequestLength="10240" executionTimeout="120" />
  </system.web>

  <system.webServer>
    <security>
      <requestFiltering>
        <!-- The content length is in bytes  -->
        <requestLimits maxAllowedContentLength="10485760"/>
      </requestFiltering>
    </security>
  </system.webServer>
</configuration>
```

نظرات خوانندگان

نویسنده: جوادنب
تاریخ: ۱۳۹۳/۱۲/۲۳ ۱۰:۸

سلام؛ یک سوال دارم در مورد متا دیتا اضافی. می‌خواستم ببینم اگر من بخوام Id یک رکورد را برای ویرایش به سمت سرور بفرستم در قسمت codeId چطوری مقدار دهی کنم. واضح‌تر بگم من یک گرید دارم که میخوام وقتی کاربر عکس جدید آپلود کرد شماره رکورد مورد نظر را داشته باشم چطوری می‌تونم CodeId آن را به صورت پویا مقدار دهی کنم؟ ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۲/۲۳ ۱۰:۴۰

مراجعه کنید به مطلب تکمیلی « استفاده از ویجت آپلود KendoUI بصورت پاپ آپ » و قسمت e.data = { Id: } آن.
options.model.Id

نویسنده: شروین ایرانی
تاریخ: ۱۳۹۴/۰۲/۱۳ ۱۱:۵۹

سلام.بی زحمت کداین اموزش را میشه قرارداد؟ مرسی

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۱۳ ۱۲:۸

مثال 9 هست.

نویسنده: شروین ایرانی
تاریخ: ۱۳۹۴/۰۲/۱۳ ۱۳:۲۲

برای اینکه قبل از حذف فایل از روی سرور تاییدیه از کاربر بگیریم باید چیکار کرد؟ من کد زیر را نوشتم اما بعد از حذف سئوال می‌پرسه؟

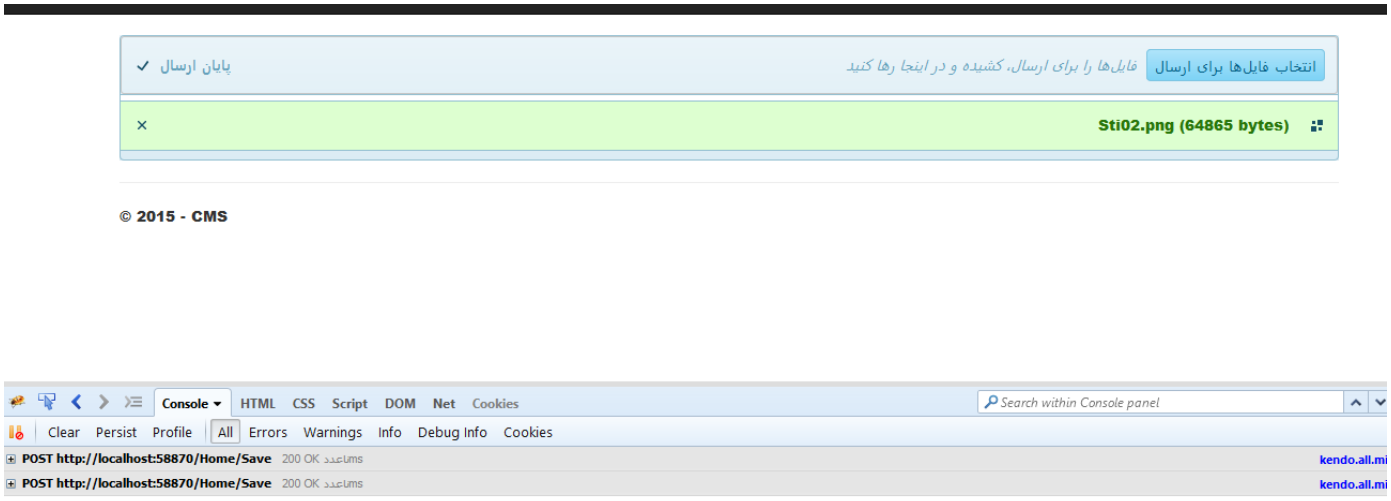
```
remove: function () {
                    var r = confirm("برای  حذف  اطمینان دارید");
                    if (r == true)
                    {
                        alert("فایل حذف شد");
                    }
                    else
                    {
                    }
                },
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۱۳ ۱۳:۵۵

```
remove: function (e) {
  if(!confirm("Are you sure you want to remove the item?")){
          e.preventDefault();
   }
},
```

نویسنده: شروین ایرانی
تاریخ: ۱۳۹۴/۰۲/۱۳ ۲۰:۴

من زمانیکه فایلی رو قصد آپلود دارم به ازاء هرفایل 2 بار درخواست به سرور ارسال میشود. مانند عکس . ممنونم اگه دلیلشو بفرمایید.(توجه به FireBug)



نویسنده: شروین ایرانی
تاریخ: ۱۳۹۴/۰۲/۱۳ ۲۰:۹

چگونه میتوان به KendoUI Uploader آدرس پوشه‌ای را داد و محتویات داخل آن‌را نمایش داد و بتوان حذف و هم اضافه نمود؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۱۴ ۱:۲۸

« Kendo UI Upload Control With Existing Files »

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۱۴ ۱:۳۶

احتمالا ویجت آپلود چندین بار در صفحه قرار گرفته‌است. قبل از شروع آن، یکبار موارد قبلی را تخریب کنید:

```
<script>
$(function() {
  kendo.destroy("#files");
});
</script>
```

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۴/۰۳/۲۵ ۱:۱۸

سلام و با تشکر؛ یک سوال:
آیا امکان دسترسی به لیست فایل‌ها در حالت عدم استفاده از batch وجود دارد؟ به عنوان مثال حالتی را در نظر بگیرید که کاربر قرار است یک فرم را تکمیل کرده در این فرم یکی از فیلدها تصاویر می‌باشد (تصاویر درون جدول دیگری ذخیره خواهند شد) یعنی اکشن متد به اینصورت باشد:

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create(MyModel model, IEnumerable<HttpPostedFileBase> files)
{
            // code
            return View();
}
```

در اینحالت مقدار دریافتی files نال خواهد بود.
به عنوان راه‌حل جایگزین از روش " **فعال سازی ارسال batch** " استفاده کردم یعنی به این صورت:

```
[HttpPost]
        public ActionResult Save(MyModel model, IEnumerable<HttpPostedFileBase> files)
        {

                var sModel = new MyViewModel
                {
                    MainPhoto = model.MainPhoto,
                    Phone = model.Phone,
                    Title = model.Title,
                    WebSiteUrl = model.WebSiteUrl,
    // ....
                };
                if (files != null)
                {
                    var images = files.Select(file => new Image
                    {
                        FileName = Helpers.Helper.UploadFile(file)
                    }).ToList();
                    sModel.Images = images;
                }
                _locationService.AddLocation(sModel);
                _uow.SaveAllChanges();

            return RedirectToAction("List");
        }
```

حالت فوق به خوبی کار میکند اما مشکل آن عدم نمایش قسمت لیست فایل‌ها و همچنین قسمت درصد پیشرفت فایل است.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۲۵ ۱:۴۳

در قسمت «فعال سازی ارسال batch » انتهای بحث توضیح دادم.
- خیر. اگر حالت batch فعال نباشد، به ازای هر فایل، این اکشن متد یکبار فراخوانی می‌شود .
+ نام files بر اساس انطباق با نام "name: "files سمت کاربر در این مثال انتخاب شده‌است و اگر این نام را تغییر دادید، نیاز است در سمت سرور هم آن‌را تغییر دهید تا نال دریافت نکنید.
- بله. نحوه‌ی طراحی آن در حالت batch به همین شکل است. درصد پیشرفت اینبار، درصد پیشرفت کلی خواهد بود.
- امضای اکشن متد دریافت فایل در سمت سرور، یک چنین خروجی باید داشته باشد:

```
// Return an empty string to signify success
return Content("");
```

نویسنده: رنجبریان
تاریخ: ۱۳۹۴/۰۶/۰۲ ۱۱:۰

با سلام ؛ من یک مشکل در هنگام ثبت فایل در دیتابس دارم که موقع ثبت چند فایل پشت سر هم در هنگام ثبت کوئری ایی که اطلاعات از دیتابیس هنگام ثبت میگیرند درست عمل نمیکنند و فقط بار اول اعمال می‌شوند

```
public ActionResult EditSave(IEnumerable<HttpPostedFileBase> files, int referid)
       {
           // OASEntities db = new OASEntities();
           // The Name of the Upload component is "files"
           if (files != null)
           {


               int i =(int)db.letterdatas.Where(x => x.referletid == referid).Max(p => p.seq);
               foreach (var file in files)
               {
                   bool failtestsize = false;
```

```
                    if (file.ContentLength > 5096000) failtestsize = true;
                    if (failtestsize) return Content(new Exception("خطا : حجم فایل ارسالی بیش از حد
مجاز می‌باشد").Message);
                  //  System.Threading.Thread.Sleep(2000);
                    var xi = db.letterdatas.Where(x => x.referletid == referid).Count();
                    if (xi > 4) return Content(new Exception("خطا : تعداد فایلهای ارسالی بیش از حد مجاز
می‌باشد").Message);


                    i++;


                    var fileName = Path.GetFileName(file.FileName);



                    byte[] filedata = new byte[file.ContentLength];
                    Stream st = file.InputStream;
                    st.Read(filedata, 0, file.ContentLength);
                    var letterdata = new letterdata
                    {
                        data = filedata,
                        extention = Path.GetExtension(file.FileName),
                        filename = fileName,
                        seq = i,
                        referletid = referid,
                        templetguid = null,
                        fileid = Guid.NewGuid(),
                        userid = db.users.Where(x => x.username == User.Identity.Name).Select(x =>
x.userid).FirstOrDefault()


                    };
                    db.letterdatas.Add(letterdata);
                    db.SaveChanges();

                }

            }

            return Content("");
        }
```

تو اکشن بالا فایل از kendo upload دریافت و ثبت می‌شه و لی موقعی که چند فایل ارسال میشه دو کوئری زیر درست عمل نمی‌کنند

```
int i =(int)db.letterdatas.Where(x => x.referletid == referid).Max(p => p.seq);
var xi = db.letterdatas.Where(x => x.referletid == referid).Count();
```

یعنی سرعت ثبت بالاتر از حدی که کوئری‌ها اطلاعات رو از db برای هر ثبت فایل بگیرند ضمنا kendo upload به صورت زیر تنظیم شده بدون ارسال همزمان batch غیرفعال است

```
@(Html.Kendo().Upload()
            .Name("files")
            .ShowFileList(true)
            .Events(x => x.Select("onxselect").Upload("onxupload").Error("o  nxerror"))
          //  .Multiple(false)
            .HtmlAttributes(
        new
        {
            accept =
                ".pdf"
        })
            .Messages(x => x.Select("انتخاب فایل‌های نامه"))
            .Messages(x => x.Remove("حذف فایل"))
             .Messages(x => x.Cancel("لغو فایل"))
            .Messages(x => x.Retry("دوباره"))
              .Messages(x => x.UploadSelectedFiles("در حال ارسال فایلها ..."))
              .Messages(x => x.HeaderStatusUploaded("کلیه فایلها ارسال شد"))
             .Messages(x => x.DropFilesHere("فایل‌ها را به اینجا بکشید"))
            .Async(a => a
                .Save("EditSave", "Lettersdata", new { referid = @ViewBag.referid })
               .Remove("EditRemove", "Lettersdata", new { referid = @ViewBag.referid })
                .AutoUpload(true)

               // .Batch(true)
            )
```

```
                    .Files(files =>
                    {
                        foreach (var f in @ViewBag.files)
                        {
                            files.Add().Name(f.filename);


                        }
                    }
                        )
                        )
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۶/۰۲ ۱۱:۳۷

- « نمایش حداکثر اندازه مجاز فایل قابل آپلود به کاربر، در ASP.Net »

- اگر قرار هست تعدادی read و write از و به بانک اطلاعاتی به صورت یک atomic operation عمل کنند، باید از تراکنش‌ها به صورت صریح استفاده کنید:

```
using (var transaction = new TransactionScope(TransactionScopeOption.Required,
 new TransactionOptions { IsolationLevel = IsolationLevel.ReadCommitted }))
{
    var database = new DatabaseContext();

    // DBC: BEGIN TRANSACTION

    var userA = database.Users.Find(1);
    var userB = database.Users.Find(2);
    userA.Name = "Admin";

    database.SaveChanges();

    userB.Age = 28;
    database.SaveChanges();


    // DBC: COMMIT TRANSACTION

    transaction.Complete();
}
```

اطلاعات بیشتر

« اصول پایگاه داده - تراکنش ها »

« آشنایی با TransactionScope »

« مفهوم READ_COMMITTED_SNAPSHOT در EF 6 »

« بررسی Transactions و Locks در SQL Server »