

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 صرفاً 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، صرفاً 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:

انتخاب فایل‌ها برای ارسال
فایل‌ها را برای ارسال، کشیده و در اینجا رها کنید

×	itextsharp.dll	✖
×	itextsharp.pdfa.dll	✖
×	itextsharp.xmlworker.dll	✖
×	PdfRpt.dll	✖
×	PdfRpt.XML	✖

ارسال فایل‌ها

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

پس از ارسال فایل‌ها به سرور:

انتخاب فایل‌ها برای ارسال		فایل‌ها را برای ارسال، کشیده و در اینجا رها کنید	پایان ارسال ✓
100%	×	itextsharp.dll	✖
100%	×	itextsharp.pdfa.dll	✖
100%	×	itextsharp.xmlworker.dll	✖
100%	×	PdfRpt.dll	✖
100%	×	PdfRpt.XML	✖

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: "خطا",
      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.');
```

### 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 عملاً 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>
```