عنوان: **فعال سازی قسمت آپلود تصویر و فایل Kendo UI Editor**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۱ ۱۶:۲۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Kendo UI

یکی دیگر از ویجت‌های Kendo UI یک HTML Editor کامل است به همراه امکانات ارسال فایل، تصویر و ... پشتیبانی از راست به چپ. در ادامه قصد داریم نحوه‌ی مدیریت نمایش لیست فایل‌ها، افزودن و حذف آن‌ها را از طریق این ادیتور بررسی کنیم.

تنظیمات ابتدایی Kendo UI Editor

در ذیل کدهای سمت کاربر فعال سازی مقدماتی Kendo UI را مشاهده می‌کنید. در قسمت tools آن، لیست امکانات و نوار ابزار مهیای آن درج شده‌اند.
دو مورد insertImage و insertFile آن نیاز به تنظیمات سمت کاربر و سرور بیشتری دارند.

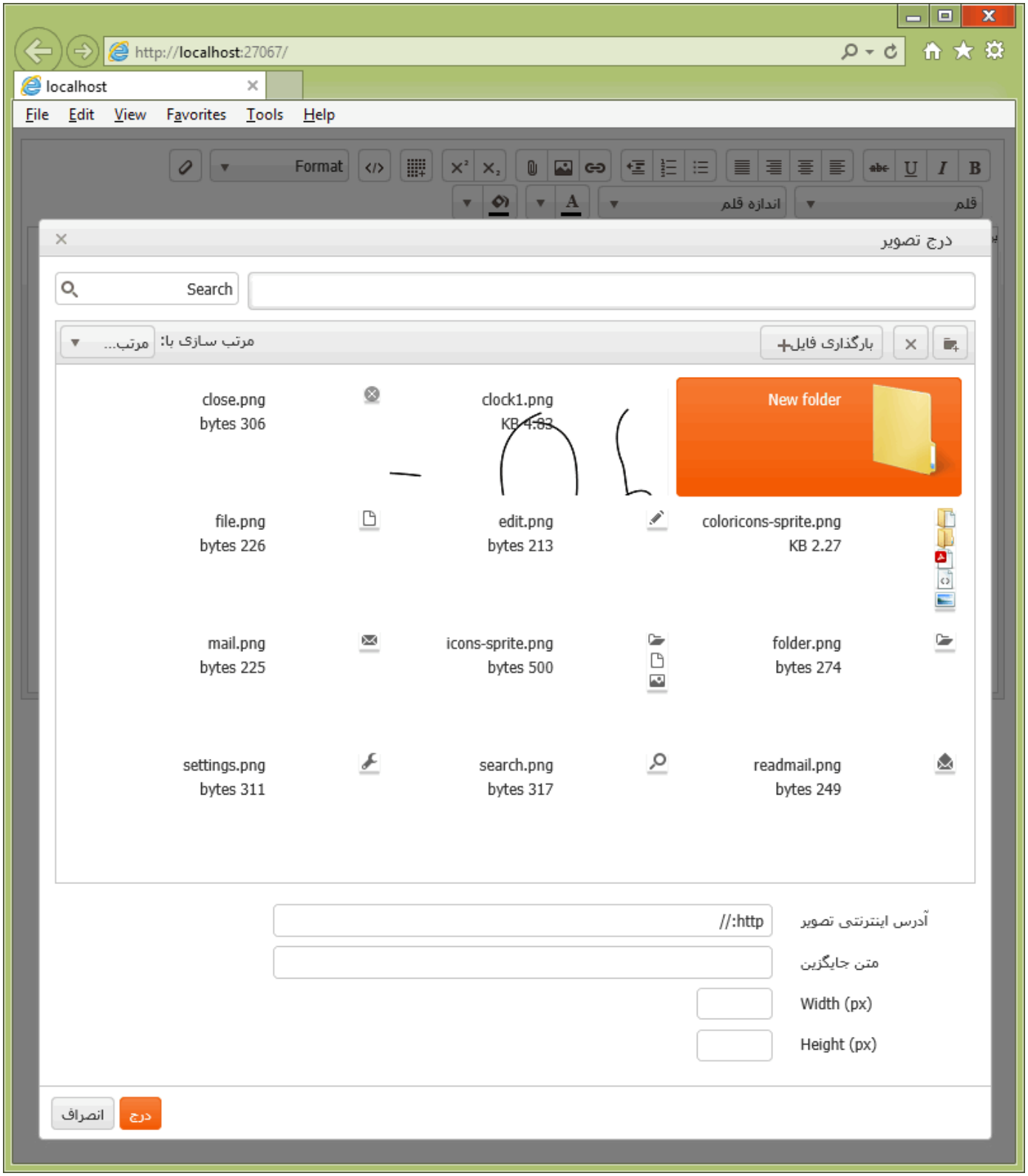
```
<!-- نحوه‌ی راست به چپ سازی -->
<div class="k-rtl">
    <textarea id="editor" rows="10" cols="30" style="height: 440px"></textarea>
</div>

@section JavaScript
{
    <script type="text/javascript">
        $(function () {
            $("#editor").kendoEditor({
                tools: [
                    "bold", "italic", "underline", "strikethrough", "justifyLeft",
                    "justifyCenter", "justifyRight", "justifyFull", "insertUnorderedList",
                    "insertOrderedList", "indent", "outdent", "createLink", "unlink",
                    "insertImage", "insertFile",
                    "subscript", "superscript", "createTable", "addRowAbove", "addRowBelow",
                    "addColumnLeft", "addColumnRight", "deleteRow", "deleteColumn", "viewHtml",
                    "formatting", "cleanFormatting", "fontName", "fontSize", "foreColor",
                    "backColor", "print"
                ],
                imageBrowser: {
                    messages: {
                        dropFilesHere: "فایل‌های خود را به اینجا کشیده و رها کنید"
                    },
                    transport: {
                        read: {
                            url: "@Url.Action("GetFilesList", "KendoEditorImages")",
                            dataType: "json",
                            contentType: 'application/json; charset=utf-8',
                            type: 'GET',
                            cache: false
                        },
                        destroy: {
                            url: "@Url.Action("DestroyFile", "KendoEditorImages")",
                            type: "POST"
                        },
                        create: {
                            url: "@Url.Action("CreateFolder", "KendoEditorImages")",
                            type: "POST"
                        },
                        thumbnailUrl: "@Url.Action("GetThumbnail", "KendoEditorImages")",
                        uploadUrl: "@Url.Action("UploadFile", "KendoEditorImages")",
                        imageUrl: "@Url.Action("GetFile", "KendoEditorImages")?path={0}"
                    }
                },
                fileBrowser: {
                    messages: {
                        dropFilesHere: "فایل‌های خود را به اینجا کشیده و رها کنید"
                    },
                    transport: {
                        read: {
                            url: "@Url.Action("GetFilesList", "KendoEditorFiles")",
                            dataType: "json",
                            contentType: 'application/json; charset=utf-8',
                            type: 'GET',
                            cache: false
                        },
                        destroy: {
```

```
                            url: "@Url.Action("DestroyFile", "KendoEditorFiles")",
                            type: "POST"
                        },
                        create: {
                            url: "@Url.Action("CreateFolder", "KendoEditorFiles")",
                            type: "POST"
                        },
                        uploadUrl: "@Url.Action("UploadFile", "KendoEditorFiles")",
                        fileUrl: "@Url.Action("GetFile", "KendoEditorFiles")?path={0}"
                    }
                }
            });
        });
    </script>
}
```

در اینجا نحوه‌ی تنظیم مسیرهای مختلف ارسال فایل و تصویر Kendo UI Editor را ملاحظه می‌کنید.
منهای قسمت thumbnailUrl، عملکرد قسمت‌های مختلف افزودن فایل و تصویر این ادیتور یکسان هستند. به همین جهت می‌توان برای مثال کنترلی مانند KendoEditorFilesController را ایجاد و سپس در کنترلر KendoEditorImagesController از آن ارث بری کرد و متد دریافت و نمایش بند انگشتی تصاویر را افزود. به این ترتیب دیگر نیازی به تکرار کدهای مشترک بین این دو قسمت نخواهد بود.

نمایش لیست پوشه‌ها و تصویر در ابتدای باز شدن صفحه‌ی درج تصویر

با کلیک بر روی دکمه‌ی نمایش لیست تصاویر، صفحه دیالوگی مانند شکل زیر ظاهر خواهد شد:



تنظیمات خواندن این فایل‌ها، از قسمت read مربوط به imageBrowser دریافت می‌شود که cache آن نیز به false تنظیم شده‌است تا در این بین مرورگر اطلاعات را کش نکند. این مورد در حین حذف فایل‌ها و پوشه‌ها مهم است. زیرا اگر cache:false تنظیم نشده باشد، حذف یک فایل یا پوشه در سمت کاربر تاثیری نخواهد داشت.

```
imageBrowser: {
    transport: {
        read: {
            url: "@Url.Action("GetFilesList", "KendoEditorImages")",
```

```
            dataType: "json",
            contentType: 'application/json; charset=utf-8',
            type: 'GET',
            cache: false
        }
    }
},
```

در ادامه نیاز است اکشن متد GetFilesList را به نحو ذیل در سمت سرور تهیه کرد:

```
namespace KendoUI13.Controllers
{
    public class KendoEditorFilesController : Controller
    {
        //مسیر پوشه فایل‌ها
        protected string FilesFolder = "~/files";

        protected string KendoFileType = "f";
        protected string KendoDirType = "d";

        [HttpGet]
        public ActionResult GetFilesList(string path)
        {
            path = GetSafeDirPath(path);
            var imagesList = new DirectoryInfo(path)
                                .GetFiles()
                                .Select(fileInfo => new KendoFile
                                {
                                    Name = fileInfo.Name,
                                    Size = fileInfo.Length,
                                    Type = KendoFileType
                                }).ToList();

            var foldersList = new DirectoryInfo(path)
                                .GetDirectories()
                                .Select(directoryInfo => new KendoFile
                                {
                                    Name = directoryInfo.Name,
                                    Type = KendoDirType
                                }).ToList();

            return new ContentResult
            {
                Content = JsonConvert.SerializeObject(imagesList.Union(foldersList), new
JsonSerializerSettings
                {
                    ContractResolver = new CamelCasePropertyNamesContractResolver()
                }),
                ContentType = "application/json",
                ContentEncoding = Encoding.UTF8
            };
        }


        protected string GetSafeDirPath(string path)
        {
            // path = مسیر زیر پوشه‌ی وارد شده
            if (string.IsNullOrWhiteSpace(path))
            {
                return Server.MapPath(FilesFolder);
            }

            //تمیز سازی امنیتی
            path = Path.GetDirectoryName(path);
            path = Path.Combine(Server.MapPath(FilesFolder), path);
            return path;
        }
    }
}
```

در اینجا کدهای کلاس پایه KendoEditorFilesController را مشاهده می‌کنید. به این جهت فیلد FilesFolder آن protected تعریف شده‌است تا در کلاسی که از آن ارث بری می‌کند نیز قابل دسترسی باشد. سپس لیست فایل‌ها و پوشه‌های path دریافتی با فرمت لیستی از KendoFile تهیه شده و با فرمت JSON بازگشت داده می‌شوند. ساختار KendoFile را در ذیل مشاهده می‌کنید:

```
namespace KendoUI13.Models
{
    public class KendoFile
    {
        public string Name { set; get; }
        public string Type { set; get; }
        public long Size { set; get; }
    }
}
```

- در اینجا Type می‌تواند از نوع فایل با مقدار f و یا از نوع پوشه با مقدار d باشد.
- علت استفاده از CamelCasePropertyNamesContractResolver در حین بازگشت JSON نهایی، تبدیل خواص دات نتی، به نام‌های سازگار با JavaScript است. برای مثال به صورت خودکار Name را تبدیل به name می‌کند.
- پارامتر path در ابتدای کار خالی است. اما کاربر می‌تواند در بین پوشه‌های باز شده‌ی توسط مرورگر تصاویر Kendo UI حرکت کند. به همین جهت مقدار آن باید هربار بررسی شده و بر این اساس لیست فایل‌ها و پوشه‌های جاری بازگشت داده شوند.

مدیریت حذف تصاویر و پوشه‌ها

همانطور که در شکل فوق نیز مشخص است، با انتخاب یک پوشه یا فایل، دکمه‌ای با آیکن ضربدر جهت فراهم آوردن امکان حذف، ظاهر می‌شود. این دکمه متصل است به قسمت destroy تنظیمات ادیتور:

```
imageBrowser: {
    transport: {
        destroy: {
            url: "@Url.Action("DestroyFile", "KendoEditorImages")",
            type: "POST"
        }
    }
},
```

این تنظیمات سمت کاربر را باید به نحو ذیل در سمت سرور مدیریت کرد:

```
namespace KendoUI13.Controllers
{
    public class KendoEditorFilesController : Controller
    {
        //مسیر پوشه فایل‌ها
        protected string FilesFolder = "~/files";

        protected string KendoFileType = "f";
        protected string KendoDirType = "d";

        [HttpPost]
        public ActionResult DestroyFile(string name, string path)
        {
            //تمیز سازی امنیتی
            name = Path.GetFileName(name);
            path = GetSafeDirPath(path);

            var pathToDelete = Path.Combine(path, name);

            var attr = System.IO.File.GetAttributes(pathToDelete);
            if ((attr & FileAttributes.Directory) == FileAttributes.Directory)
            {
                Directory.Delete(pathToDelete, recursive: true);
            }
            else
            {
                System.IO.File.Delete(pathToDelete);
            }

            return Json(new object[0]);
        }
    }
}
```

- استفاده از Path.GetFileName جهت دریافت نام فایل‌ها در اینجا بسیار مهم است. زیرا اگر این تمیز سازی امنیتی صورت نگیرد، ممکن است با کمی تغییر در آن، فایل web.config برنامه، دریافت یا حذف شود.
- پارامتر name دریافتی مساوی است با نام فایل انتخاب شده و path مشخص می‌کند که در کدام پوشه قرار داریم.
- چون در اینجا امکان حذف یک پوشه یا فایل وجود دارد، حتما نیاز است بررسی کنیم، مسیر دریافتی پوشه‌است یا فایل و سپس بر این اساس جهت حذف آن‌ها اقدام صورت گیرد.

مدیریت ایجاد یک پوشه‌ی جدید

تنظیمات قسمت create مرورگر تصاویر، مرتبط است به زمانیکه کاربر با کلیک بر روی دکمه‌ی +، درخواست ایجاد یک پوشه‌ی جدید را کرده‌است:

```
imageBrowser: {
    transport: {
        create: {
            url: "@Url.Action("CreateFolder", "KendoEditorImages")",
            type: "POST"
        }
    }
},
```

کدهای اکشن متد متناظر با این عمل را در ذیل مشاهده می‌کنید:

```
namespace KendoUI13.Controllers
{
    public class KendoEditorFilesController : Controller
    {
        //مسیر پوشه فایل‌ها
        protected string FilesFolder = "~/files";

        protected string KendoFileType = "f";
        protected string KendoDirType = "d";

        [HttpPost]
        public ActionResult CreateFolder(string name, string path)
        {
            //تمیز سازی امنیتی
            name = Path.GetFileName(name);
            path = GetSafeDirPath(path);
            var dirToCreate = Path.Combine(path, name);

            Directory.CreateDirectory(dirToCreate);

            return KendoFile(new KendoFile
            {
                Name = name,
                Type = KendoDirType
            });
        }

        protected ActionResult KendoFile(KendoFile file)
        {
            return new ContentResult
            {
                Content = JsonConvert.SerializeObject(file,
                    new JsonSerializerSettings
                    {
                        ContractResolver = new CamelCasePropertyNamesContractResolver()
                    }),
                ContentType = "application/json",
                ContentEncoding = Encoding.UTF8
            };
        }
    }
}
```

- در اینجا نیز name مساوی نام پوشه‌ی درخواستی است و path به مسیر تو در توی پوشه‌ی جاری اشاره می‌کند.
- پس از ایجاد پوشه، باید نام آن‌را با فرمت KendoFile به صورت JSON بازگشت داد. همچنین در اینجا Type را نیز باید به d

(پوشه) تنظیم کرد.

مدیریت قسمت ارسال فایل و تصویر

زمانیکه کاربر بر روی دکمه‌ی upload file یا بارگذاری تصاویر در اینجا کلیک می‌کند، اطلاعات فایل آپلودی به مسیر uploadUrl ارسال می‌گردد.

```
imageBrowser: {
    transport: {
        thumbnailUrl: "@Url.Action("GetThumbnail", "KendoEditorImages")",
        uploadUrl: "@Url.Action("UploadFile", "KendoEditorImages")",
        imageUrl: "@Url.Action("GetFile", "KendoEditorImages")?path={0}"
    }
},
```

دو تنظیم دیگر thumbnailUrl و imageUrl، برای نمایش بند انگشتی و نمایش کامل تصویر کاربرد دارند.
در ادامه کدهای مدیریت سمت سرور قسمت آپلود این ادیتور را مشاهده می‌کنید:

```
namespace KendoUI13.Controllers
{
    public class KendoEditorFilesController : Controller
    {
        //مسیر پوشه فایل‌ها
        protected string FilesFolder = "~/files";

        protected string KendoFileType = "f";
        protected string KendoDirType = "d";


        [HttpPost]
        public ActionResult UploadFile(HttpPostedFileBase file, string path)
        {
            //تمیز سازی امنیتی
            var name = Path.GetFileName(file.FileName);
            path = GetSafeDirPath(path);
            var pathToSave = Path.Combine(path, name);

            file.SaveAs(pathToSave);

            return KendoFile(new KendoFile
            {
                Name = name,
                Size = file.ContentLength,
                Type = KendoFileType
            });
        }
    }
}
```

- در اینجا path مشخص می‌کند که در کدام پوشه‌ی تو در تو قرار داریم و file نیز حاوی محتوای ارسالی به سرور است.
- پس از ذخیره سازی اطلاعات فایل، نیاز است اطلاعات فایل نهایی را با فرمت KendoFile به صورت JSON بازگشت دهیم.

ارث بری از KendoEditorFilesController جهت تکمیل قسمت مدیریت تصاویر

تا اینجا کدهایی را که ملاحظه کردید، برای هر دو قسمت ارسال تصویر و فایل کاربرد دارند. قسمت ارسال تصاویر برای تکمیل نیاز به متد دریافت تصاویر به صورت بند انگشتی نیز دارد که به صورت ذیل قابل تعریف است و چون از کلاس پایه KendoEditorFilesController ارث بری کرده‌است، این کنترلر به صورت خودکار حاوی اکشن متدهای کلاس پایه نیز خواهد بود.

```
using System.Web.Mvc;

namespace KendoUI13.Controllers
{
    public class KendoEditorImagesController : KendoEditorFilesController
```

```
    {
        public KendoEditorImagesController()
        {
            // بازنویسی مسیر پوشه‌ی فایل‌ها
            FilesFolder = "~/images";
        }

        [HttpGet]
        [OutputCache(Duration = 3600, VaryByParam = "path")]
        public ActionResult GetThumbnail(string path)
        {
            //todo: create thumb/ resize image

            path = GetSafeFileAndDirPath(path);
            return File(path, "image/png");
        }
    }
}
```

کدهای کامل این مطلب را از اینجا می‌توانید دریافت کنید.

نظرات خوانندگان

نویسنده: حسین صفدری
تاریخ: ۱۳۹۴/۰۱/۲۲ ۱۰:۵۸

باسلام
ممنون بابت این مقاله..
من این رو چند وقت پیش امتحان کردم اما مشکل توی ویرایش بود، مثلا تصویر لود نمی‌شد(به صورت لینک نمایش داده می‌شد.)، ضاهر به هم می‌ریخت که دوباره مجبور شدم همون tiny mce رو امتحان کنم...
شما و دوستان طبق تجربه کدوم رو پیشنهاد می‌کنند؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۲ ۱۱:۳۰

بارگذاری تصویر در این ادیتور بر اساس تنظیم پارامتر imageUrl در سمت کلاینت انجام می‌شود. یعنی تصویر قرار گرفته در ادیتور برای ذخیره‌ی نهایی، src تگ img آن بر اساس آدرس و پارامتر imageUrl درج خواهد شد (و نه یک آدرس مستقیم) که معادل است با اکشن متد زیر در سمت سرور:

```
[HttpGet]
public ActionResult GetFile(string path)
{
    path = GetSafeFileAndDirPath(path);
    return File(path, "image/png");
}
```

این روش در سایر ادیتورها هم بکار رفته‌است (^ و ^).
البته مثال فوق آزمایش شده‌است و در این بین از روش دیباگ اسکریپت‌ها و یافتن خطاهای سمت سرور و کاربر کمک گرفته شد (کار کردن با کتابخانه‌های جاوا اسکریپتی، بدون باز نگه داشتن کنسول developer مرورگرها تقریبا غیر ممکن است).

نویسنده: غلامرضا ربال
تاریخ: ۱۳۹۴/۰۱/۲۴ ۱۳:۱۳

سلام.
آیا این امکان وجود دارد از قسمت ImageBrowser , FileBrowser به صورت یک Widget خودکفا و بدون اینکه با ویرایشگر متن یکپارچه شود ،استفاده کرد؟
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۴ ۱۴:۳۸

خیر. ولی با توجه به سورس آن (kendo.editor.zip) نحوه‌ی تشکیل این نمایشگرها با استفاده از ترکیب kendo.template و همچنین ویجت kendoWindow است.

نویسنده: غلامرضا ربال
تاریخ: ۱۳۹۴/۰۱/۲۴ ۱۵:۴

در انجمن تلریک چنین بحثی شده بود و پاسخی که داده شده بود به شکل زیر است:

```
<div id="imgBrowser"></div>

$("#imgBrowser").kendoImageBrowser({
    transport: {
        read: "/service/ImageBrowser/Read",
        destroy: {
```

```
            url: "/service/ImageBrowser/Destroy",
            type: "POST"
        },
        create: {
            url: "/service/ImageBrowser/Create",
            type: "POST"
        },
        thumbnailUrl: "/service/ImageBrowser/Thumbnail",
        uploadUrl: "/service/ImageBrowser/Upload",
        imageUrl: "/service/ImageBrowser/Image?path={0}"
    }
});
```

بنده خودم هم تست کردم .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۴ ۱۵:۴۲

بله. فایل‌های مجزایی به نام‌های kendo.filebrowser.js و kendo.imagebrowser.js (با این سورس‌ها kendo.imagebrowser.zip و kendo.filebrowser.zip) در این مجموعه وجود دارند ولی خارج از ادیتور جایی از آن‌ها استفاده نشده و مستندات رسمی هم ندارند.