عنوان: Ajax.BeginForm و ارسال فایل به سرور در ASP.NET MVC
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۱ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, jQuery Ajax, File upload

Ajax.BeginForm در ASP.NET MVC از jQuery Ajax برای ارسال مقادیر فرم، به سرور استفاده می‌کند. در این بین اگر یکی از عناصر فرم، المان ارسال فایل به سرور باشد، مقدار دریافتی در سمت سرور نال خواهد بود. مشکل اینجا است که نمی‌توان به کمک Ajax معمولی (یا به عبارتی XMLHttpRequest) فایلی را به سرور ارسال کرد. یا باید از سیلورلایت یا فلش استفاده نمود و یا از مرورگرهایی که XMLHttpRequest Level 2 را پشتیبانی می‌کنند (از IE 10 به بعد مثلا) که امکان Ajax upload توکار به همراه گزارش درصد آپلود را بدون نیاز به فلش یا سیلورلایت، دارند.
در این بین راه حل دیگری نیز وجود دارد که با تمام مرورگرها سازگار است؛ اما تنها گزارش درصد آپلود را توسط آن نخواهیم داشت. در اینجا به صورت پویا یک IFrame مخفی در صفحه تشکیل می‌شود، مقادیر معمولی فرم (تمام المان‌ها، منهای file) به صورت Ajax ایی به سرور ارسال خواهند شد. المان file آن در این IFrame مخفی، به صورت معمولی به سرور Postback می‌شود. البته کاربر در این بین چیزی را مشاهده یا احساس نخواهد کرد و تمام عملیات از دیدگاه او Ajax ایی به نظر می‌رسد.
برای انجام اینکار تنها کافی است از افزونه‌ی AjaxFileUpload استفاده کنیم که در ادامه نحوه‌ی استفاده از آن‌را بررسی خواهیم کرد.

پیشنیازها

در ادامه فرض بر این است که افزونه‌ی AjaxFileUpload را دریافت کرده و به فایل Layout برنامه افزوده‌اید:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
    <link href="~/Content/Site.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <div>
        @RenderBody()
    </div>

    <script src="~/Scripts/jquery-1.11.1.min.js"></script>
    <script src="~/Scripts/jquery.unobtrusive-ajax.js"></script>
    <script src="~/Scripts/ajaxfileupload.js"></script>
    @RenderSection("Scripts", required: false)
</body>
</html>
```

مدل، کنترلر و View برنامه

مدل برنامه مشخصات یک محصول است:

```
namespace MVCAjaxFormUpload.Models
{
    public class Product
    {
        public int Id { set; get; }
        public string Name { set; get; }
    }
}
```

کنترلر آن از سه متد تشکیل شده‌است:

```
using System.Threading;
using System.Web;
```

```
using System.Web.Mvc;
using MVCAjaxFormUpload.Models;

namespace MVCAjaxFormUpload.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        [HttpPost]
        public ActionResult Index(Product product)
        {
            var isAjax = this.Request.IsAjaxRequest();

            return Json(new { result = "ok" }, JsonRequestBehavior.AllowGet);
        }

        [HttpPost]
        public ActionResult UploadFiles(HttpPostedFileBase image1, int id)
        {
            var isAjax = this.Request.IsAjaxRequest();
            Thread.Sleep(3000); //شبیه سازی عملیات طولانی
            return Json(new { FileName = "/Uploads/filename.ext" }, "text/html",
JsonRequestBehavior.AllowGet);
        }
    }
}
```

Index اول، کار نمایش صفحه‌ی ارسال اطلاعات را انجام خواهد داد.

Index دوم کار پردازش Ajax ایی اطلاعات ارسالی به سرور را به عهده دارد. HttpPost آن Ajax ایی است.

متد UploadFiles، کار پردازش اطلاعات ارسالی از طرف IFrame مخفی را انجام می‌دهد. HttpPost آن معمولی است.

و کدهای View این مثال نیز به شرح زیر است:

```
@model MVCAjaxFormUpload.Models.Product
@{
    ViewBag.Title = "Index";
}

<h2>Ajax Form Upload</h2>

@using (Ajax.BeginForm(actionName: "Index",
                       controllerName: "Home",
                       ajaxOptions: new AjaxOptions { HttpMethod = "POST" },
                       routeValues: null,
                       htmlAttributes: new { id = "uploadForm" }))
{
    <label>Name:</label>
    @Html.TextBoxFor(model => model.Name)
    <br />
    <label>Image:</label>
    <br />
    <input type="file" name="Image1" id="Image1" />
    <br />
    <input type="submit" value="Submit" />
    <img id="loading" src="~/Content/Images/loading.gif" style="display:none;">
}

@section Scripts
{
    <script type="text/javascript">
        $(function () {
            $('#uploadForm').submit(function () {
                $("#loading").show();
                $.ajaxFileUpload({
                    url: "@Url.Action("UploadFiles", "Home")", // مسیری که باید فایل به آن ارسال شود
                    secureuri: false,
                    fileElementId: 'Image1', // آی دی المان ورودی فایل
                    dataType: 'json',
                    data: { id: 1, data: 'test' }, // اطلاعات اضافی در صورت نیاز
                    success: function (data, status) {
                        $("#loading").hide();
```

```
                        if (typeof (data.FileName) != 'undefined') {
                            alert(data.FileName);
                        }
                    },
                    error: function (data, status, e) {
                        $("#loading").hide();
                        alert(e);
                    }
                });
            });
        });
    </script>
}
```

فرمی که توسط Ajax.BeginForm تشکیل شده‌است، یک فرم معمولی Ajax ایی است و نکته‌ی جدیدی ندارد. تنها در آن یک المان ارسال فایل قرار گرفته‌است و همچنین Id آن‌را نیز جهت استفاده توسط jQuery مشخص کرده‌ایم.
در ادامه نحوه‌ی فعال سازی ajaxFileUpload را دقیقا در زمان submit فرم، مشاهده می‌کنید. در اینجا url آن به اکشن متدی که اطلاعات المان file را باید دریافت کند، اشاره می‌کند. fileElementId آن مساوی Id المان فایل فرم Ajax ایی صفحه‌است. از قسمت data جهت ارسال اطلاعات اضافه‌تری به اکشن متد UploadFiles استفاده می‌شود. سایر قسمت‌های آن نیز مشخص هستند. اگر عملیات موفقیت آمیز بود، success آن و اگر خیر، error آن اجرا می‌شوند.
فقط باید دقت داشت که content type دریافتی توسط آن باید text/html باشد، که این مورد در اکشن متدهای کنترلر مشخص هستند.
به این ترتیب دیگر کاربر نیازی ندارد ابتدا یکبار بر روی دکمه‌ی دومی کلیک کرده و فایل را ارسال کند و سپس بار دیگر بر روی دکمه‌ی submit فرم کلیک نماید. هر دو کار توسط یک دکمه انجام می‌شوند.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید

MVCAjaxFormUpload.zip

نظرات خوانندگان

نویسنده: M.Shakeri
تاریخ: ۱۳۹۳/۰۴/۲۲ ۱۰:۱۸

ممنون جناب نصیری بابت مطلبی که قرار دادین.
من چند ماه پیش از این افزونه استفاده کردم.میخواستم فرمی از اطلاعات رو به همراه فایل مربوطه بصورت ajax ارسال کنم اما جدول آپلود فایل من جدا بود و به ازای هر فرم کلید خارجی اطلاعات مربوطه در جدول آپلود ثبت می‌شد.
مسئله اینجا بود که من باید اول فرم رو ثبت می‌کردم و سپس کلید خارجی اون فرم رو در جدول آپلود قرار میدادم اما مشکل من این بود که اول فایل آپلود میشد و بعد فرم ثبت میشد.یعنی اول رویداد submit کلاینت اجرا میشد سپس Ajax.BeginForm.
برای این مسئله راهکاری هست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۲ ۱۲:۴۵

در حین تعریف فرم، OnSuccess را به یک متد جاوا اسکریپتی که قرار است پس از اجرای موفقیت آمیز ارسال اطلاعات Ajax ایی به سرور اجرا شود، مقدار دهی کنید:

```
@using (Ajax.BeginForm(actionName: "Index",
                       controllerName: "Home",
                       ajaxOptions: new AjaxOptions
                       {
                           HttpMethod = "POST",
                           OnSuccess = "doUpload(data, status, xhr)"
                       },
                       routeValues: null,
                       htmlAttributes: new { id = "uploadForm" }))
{
```

این متد یک چنین امضایی را باید داشته باشد:

```
function doUpload(data, status, xhr) {
            alert(data.result);
            // مابقی کدهای آپلود فایل
```

به عبارتی می‌توان Id رکورد insert شده را در اینجا دریافت (توسط data) و سپس به کمک اطلاعات اضافی ارسال به سرور افزونه‌ی ارسال فایل، به اکشن متد ذخیره فایل ارسال کرد.

جهت تکمیل بحث

- OnBegin - xhr
- OnComplete - xhr, status
- OnSuccess - data, status, xhr
- OnFailure - xhr, status, error

پارامترهای AjaxOptions یک چنین اطلاعاتی را از سرور دریافت می‌کنند که نمونه‌ای از آن در متد doUpload فوق استفاده شد.

نویسنده: رضا
تاریخ: ۱۳۹۳/۰۷/۳۰ ۱۸:۷

جناب آقای نصیری ممنون از مطلبتون
در صورتیکه ارسال فایل بصورت فرم‌های مودال مثلا jquery ui و بصورت ajax بخواد انجام بگیره و نیاز به محتوای فایل مورد نظر برای ذخیره باشه (اطلاعات فرم مودال و فایل در یک جدول ذخیره بشه) چطور میشه توی controller توی اکشن متد پست مربوطه به HttpPostedFileBase دسترسی پیدا کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۳۰ ۱۸:۳۷

تفاوتی نمی‌کند. افزونه‌ی مورد استفاده وابستگی خاصی به ASP.NET MVC ندارد. خلاصه عملیات فوق به این نحو است که فراخوانی این افزونه در زمان submit فرم انجام می‌شود. همین رویداد را در فرم‌های مودال یا هر فرم دیگری نیز می‌توان تحت کنترل قرار داد و بازنویسی کرد.

نویسنده: پویا امینی
تاریخ: ۱۳۹۳/۰۸/۳۰ ۲۱:۱۰

با سلام
من یک فرم دارم که در اون کاربر باید دو عکس وارد کنید به صورت زیر

```
<div>
        <div style="margin: 2px 4px !important">
            <div>
                1 عکس
                <input type="file" name="imageSrc" id="imageSrc" />
                @*@Html.Kendo().Upload().Name("imageSrc").Multiple(false)*@
                @Html.ValidationMessageFor(model => model.Image)
            </div>
        </div>

    </div>
    <div>
        <div style="margin: 2px 4px !important">
            <div>
                2 عکس
                <input type="file" name="imageSrc2" id="imageSrc2" />
                @*@Html.Kendo().Upload().Name("imageSrc2").Multiple(false)*@
                @Html.ValidationMessageFor(model => model.Image)
            </div>
        </div>
```

آیا می‌تونم این دو فایل رو با هم آپلود کنم؟ چون در مثال شما 'fileElementId: 'Image1 , فقط نام یک کنترل را می‌گیرد.
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۳۰ ۲۱:۵۰

اصل افزونه در اینجا مطرح شده‌است. سورس آن‌را دریافت کنید و مطابق نیاز خودتان آن‌را توسعه دهید (بجای یک fileElementId که در سورس وجود دارد، آن‌را تبدیل کنید به یک آرایه مثلا).

نویسنده: محمدرضا احمدی
تاریخ: ۱۳۹۳/۱۰/۱۶ ۱۹:۲۲

با سلام؛ در صورتی که قصد ذخیره سازی اطلاعات یک محصول را داشته باشم و یکی از Propertyهای کلاس محصول ImageName باشد، باید نام عکس ارسالی را دریافت و ذخیره کنیم، اکشن متد UpoadFiles نام عکس ذخیره شده را برگشت می‌دهد، حال به چه طریقی میتوان به این نام در اکشن متد Index دسترسی داشت؟ نام برگشتی از اکشن متد UploadFiles را در یک HiddenField قرار می‌دهم ولی باز هم در اکشن Index در دسترس نیست.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۱۶ ۱۹:۳۵

کمی بالاتر پاسخ دادم. جایی که فرم را تعریف می‌کنید، خاصیت OnSuccess = doUpload را هم مقدار دهی کنید. خروجی اکشن متد Index، اینبار Id رکورد ثبت شده را هم داشته باشد. به این Id در متد doUpload دسترسی خواهید داشت. در ادامه کدهای

$('uploadForm').submit کلا حذف می‌شوند و متد ajaxFileUpload آن به داخل متد doUpload که در حین OnSuccess فراخوانی شده، منتقل خواهد شد. در اینجا Id را به اکشن متد UploadFiles ارسال کنید (توسط خاصیت data که در مثال هست). بر مبنای این Id، رکورد را یافته و خاصیت نام فایل را به روز کنید. بنابراین به صورت خلاصه، یکبار ثبت معمولی رخ می‌دهد و Id نهایی را بازگشت خواهد داد و پس از آن در صورت OnSuccess، به صورت خودکار کار آپلود انجام خواهد شد؛ به همراه ارسال این Id به سرور.