chsakell's Blog

ANYTHING AROUND ASP.NET MVC,WEB API, WCF, ENTITY FRAMEWORK & C#

Web API File Uploading (Desktop and Web client)

BY CHRISTOS S. *on* JUNE 7, 2015 · (3)

File uploading is one of the most common tasks all developers have to deal with, at some point of their application development cycle. For example, Web applications *(such as social networks)* usually require users to upload a profile picture during registration or in the case of desktop ones, users may required to upload private documents.
This post will show you how to upload multipart/form-data files using **Web API** with both Web and Desktop clients. More particular here's what we gonna see:

- **Web API File Upload Controller**: Create an action for Uploading multipart-form data
- **Web client**: Create an AngularJS web client to upload multiple files at once using the Angular File Upload module
- **Desktop client**: Create an Windows Form client to upload multiple files using HttpClient

Let's start:

The Controller

Create a solution named *WebApiFileUpload* and add a new empty Web Application named WebApiFileUpload.API selecting both the Web API and MVC checkboxes. Add a Web API Controller named **FileUploadController**. Before showing controller's code let's see what this controller requires:

- **A FileUpload result**: Custom result to return to clients
- **A custom MultipartFormDataStreamProvider**: The provider which will actually capture and save the uploaded file
- **Allow only Mime-Multipart content to be uploaded**:For this our best option is to create a Filter and apply it to the POST method

Create a *Infrastructure* folder inside the API project and add the following three classes that

implement the above required behaviors.

FileUploadResult.cs

```
public class FileUploadResult
    {
        public string LocalFilePath { get; set; }
        public string FileName { get; set; }
        public long FileLength { get; set; }
    }
```

UploadMultipartFormProvider.cs

```
public class UploadMultipartFormProvider : MultipartFormDataStreamProvid
    {
        public UploadMultipartFormProvider(string rootPath) : base(rootP

        public override string GetLocalFileName(HttpContentHeaders heade
        {
            if (headers != null &&
                headers.ContentDisposition != null)
            {
                return headers
                    .ContentDisposition
                    .FileName.TrimEnd('"').TrimStart('"');
            }

            return base.GetLocalFileName(headers);
        }
    }
```

MimeMultipart.cs

```
public class MimeMultipart : ActionFilterAttribute
    {
        public override void OnActionExecuting(HttpActionContext actionC
        {
            if (!actionContext.Request.Content.IsMimeMultipartContent())
            {
                throw new HttpResponseException(
                    new HttpResponseMessage(
                        HttpStatusCode.UnsupportedMediaType)
                );
            }
        }

        public override void OnActionExecuted(HttpActionExecutedContext
        {

        }
    }
```

And now the controller:

FileUploadController.cs

```
public class FileUploadController : ApiController
    {
        [MimeMultipart]
        public async Task<FileUploadResult> Post()
        {
            var uploadPath = HttpContext.Current.Server.MapPath("~/Uploa

            var multipartFormDataStreamProvider = new UploadMultipartFor

            // Read the MIME multipart asynchronously
            await Request.Content.ReadAsMultipartAsync(multipartFormData

            string _localFileName = multipartFormDataStreamProvider
                .FileData.Select(multiPartData => multiPartData.LocalFil

            // Create response
            return new FileUploadResult
            {
                LocalFilePath = _localFileName,

                FileName = Path.GetFileName(_localFileName),

                FileLength = new FileInfo(_localFileName).Length
            };
        }
    }
```

Make sure you create a folder named *Uploads* inside the project cause this is where the uploaded files will be saved. The uploaded result will contain the file uploaded local path in the server, file's name and length in bytes. You can add any other properties you want.

AngularJS Web client

It's time to create a web client using AngularJS that will be able to upload multiple files using the controller we have just created. As we have mentioned we are going to use the **AngularJS File Upload** module which can be found **here**. Create a *Scripts* folder inside the API project and you need to add the following javascript files:

1. angular.js
2. angular-file-upload.js
3. angular-route.js
4. jquery-1.8.2.js *(you can use any other version if you want)*

Also add the **bootstrap.css** inside a *Styles* folder. Add a HomeController MVC Controller and add an **Index()** action if not exists. Create the respective View for this action and paste the following code:

Index.cshtml

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
    <link href="~/Styles/bootstrap.css" rel="stylesheet" />
</head>
<body ng-app="angularUploadApp">
    <div class="container">
        <div class="jumbotron">
            <p>Web API File uploading using AngularJS client. Read more
        </div>
        <div ng-include="'app/templates/fileUpload.html'"></div>

    </div>
    <script src="Scripts/jquery-1.8.2.js"></script>
    <script src="Scripts/angular.js"></script>
    <script src="Scripts/angular-file-upload.js"></script>
    <script src="Scripts/angular-route.js"></script>
    <script src="app/app.js"></script>
    <script src="app/controllers/fileUploadCtrl.js"></script>

</body>
</html>
```

We haven't created either the angularJS *fileUpload.html* or the *app.js* and *fileUploadCtrl.js* yet, so let's add them. You have to create the following folders first:

1. **app**
2. **app/controllers**
3. **app/templates**

app/app.js

```
'use strict';

angular.module('angularUploadApp', [
    'ngRoute',
    'angularFileUpload'
])

.config(function ($routeProvider) {
    $routeProvider
    .when('/', {
        templateUrl: 'app/templates/fileUpload.html',
        controller: 'fileUploadCtrl'
    })
    .otherwise({
```

```
        redirectTo: '/'
    });
});
```

app/controllers/fileUploadCtrl.js

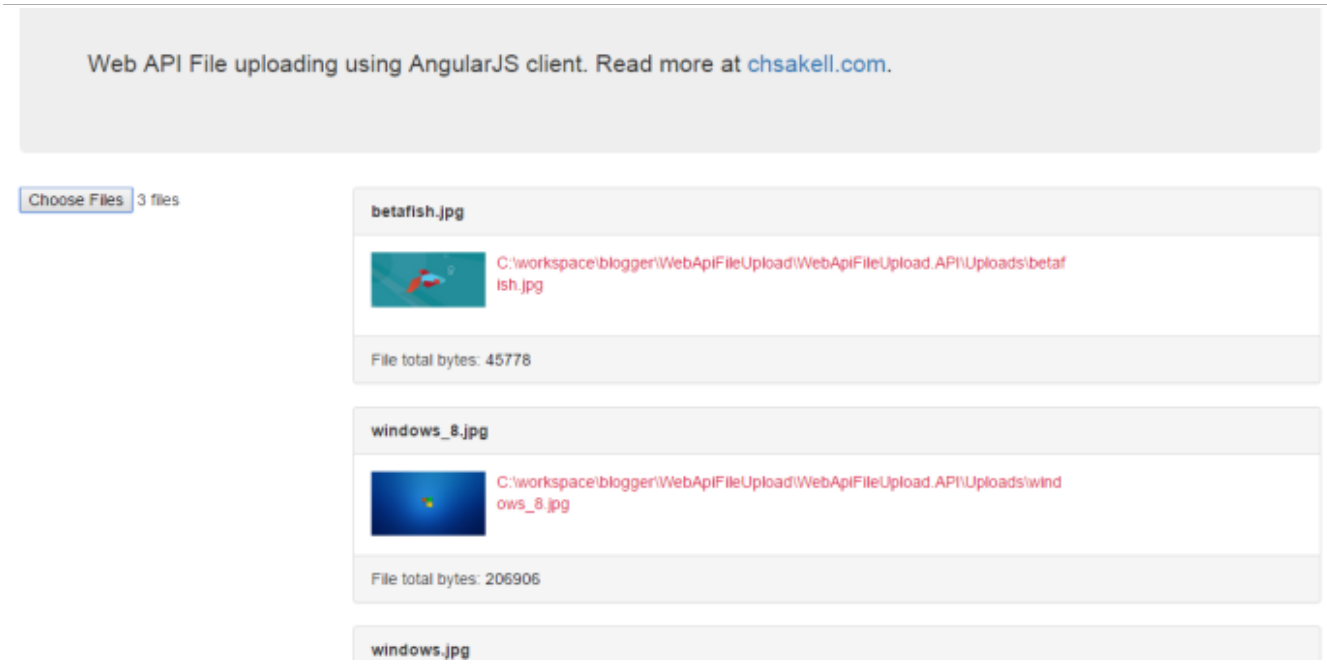
```
'use strict';

angular.module('angularUploadApp')
    .controller('fileUploadCtrl', function ($scope, $http, $timeout, $up
        $scope.upload = [];
        $scope.UploadedFiles = [];

        $scope.startUploading = function ($files) {
            //$files: an array of files selected
            for (var i = 0; i < $files.length; i++) {
                var $file = $files[i];
                (function (index) {
                    $scope.upload[index] = $upload.upload({
                        url: "./api/fileupload", // webapi url
                        method: "POST",
                        file: $file
                    }).progress(function (evt) {
                    }).success(function (data, status, headers, config)
                        // file is uploaded successfully
                        $scope.UploadedFiles.push({ FileName: data.FileN
                    }).error(function (data, status, headers, config) {
                        console.log(data);
                    });
                })(i);
            }
        }
    });
```

app/templates/fileUpload.html

```
<div class="row" ng-controller="fileUploadCtrl">
    <div class="col-xs-3">
        <div>
            <input type="file" ng-file-select="startUploading($files)" m
        </div>
    </div>
    <div class="col-xs-9">
        <div class="panel panel-default" ng-repeat="uploadedFile in Uplo
            <div class="panel-heading">
                <strong>{{uploadedFile.FileName}}</strong>
            </div>
            <div class="panel-body">
                <div class=" media">
                    <a class="pull-left" href="#">
                        <img class="media-object" width="100" ng-src="..
                    </a>
                    <div class="media-body">
                        <div class="lead" style="font-size:14px;color: c
                    </div>
                </div>
            </div>
            <div class="panel-footer">
```

```
23                 File total bytes: <span style="color:black">{{uploadedFi
24             </div>
25         </div>
26     </div>
27 </div>
```

Build and fire your application. Select multiple files to upload and ensure that all worked as expected.



Desktop Client

For the Windows Form client you need to add a new Windows Form project named *WebApiFileUpload.DesktopClient* and add reference to the *WebApiFileUpload.API* project (just to access the FileUploadResult class). You also need to install the

Microsoft.AspNet.WebApi.Client through the Nuget Packages. I have created a simple Form that allows the user to select multiple files and upload them through the API Controller we created. Let's see the main method that does that.

Part of Form1.cs

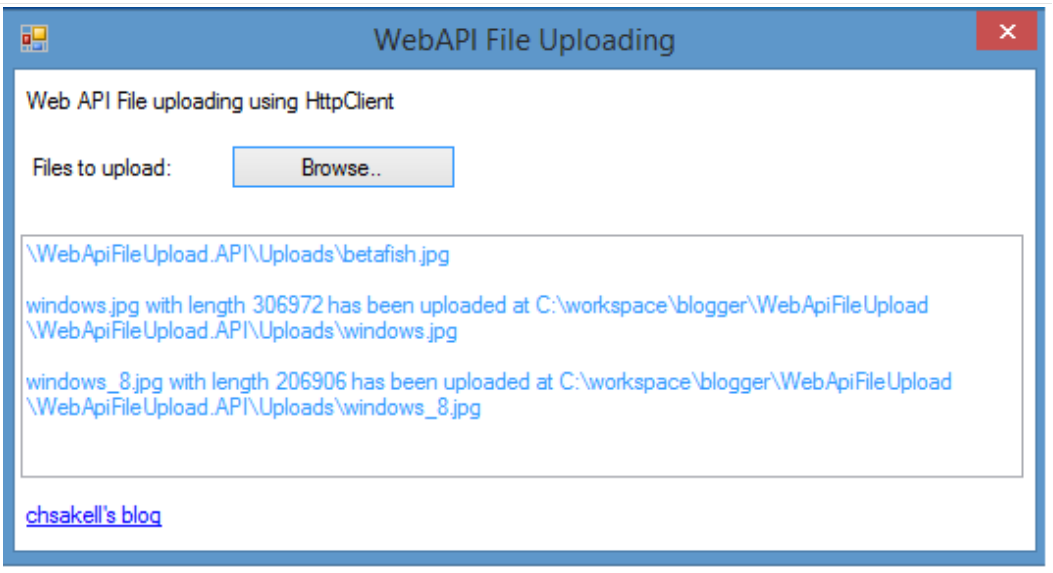
```
try
                {
                    HttpClient httpClient = new HttpClient();
                    // Read the files
                    foreach (String file in openFileDialog1.FileNames)
                    {
                        var fileStream = File.Open(file, FileMode.Open);
                        var fileInfo = new FileInfo(file);
                        FileUploadResult uploadResult = null;
                        bool _fileUploaded = false;

                        var content = new MultipartFormDataContent();
                        content.Add(new StreamContent(fileStream), "\"fi
                        );

                        Task taskUpload = httpClient.PostAsync(uploadSer
                        {
                            if (task.Status == TaskStatus.RanToCompletio
                            {
                                var response = task.Result;

                                if (response.IsSuccessStatusCode)
                                {
                                    uploadResult = response.Content.Read
                                    if (uploadResult != null)
                                        _fileUploaded = true;

                                    // Read other header values if you w
                                    foreach (var header in response.Cont
                                    {
                                        Debug.WriteLine("{0}: {1}", head
                                    }
                                }
                                else
                                {
                                    Debug.WriteLine("Status Code: {0} -
                                    Debug.WriteLine("Response Body: {0}"
                                }
                            }

                            fileStream.Dispose();
                        });

                        taskUpload.Wait();
                        if (_fileUploaded)
                            AddMessage(uploadResult.FileName + " with le
                                            + " has been uploaded at " +
                    }

                    httpClient.Dispose();
                }
```

```
52                  catch (Exception ex)
53                  {
54                      AddMessage(ex.Message);
55                  }
```

Of course you need to run the API Project first before trying to upload files using the Desktop client.

That's it, we are done! We saw how to upload multipart-form data files using Web API from both Web and Desktop clients. I hope you have enjoyed the post. You can download the project we built from **here**.

In case you find my blog's content interesting, register your email to receive notifications of new posts and follow *chsakell's Blog* on its Facebook or Twitter accounts.



3 replies

Gerson Júnior

September 9, 2015 • 1:06 am

Amazing article!

> 
>
> **defilerc**
>
> **October 21, 2015 • 8:01 am**
>
> Great article! Also, if you want to upload a file in web api but load it in the buffer instead of writing it to a file, you can also use MultipartMemoryStreamProvider instead of MultipartFormDataStreamProvider. For more info see: **http://bit.ly/1OHA3R7**

Trackbacks

1. **Web API File Uploading (Desktop and Web client) | Dinesh Ram Kali.**