WebFileUploader

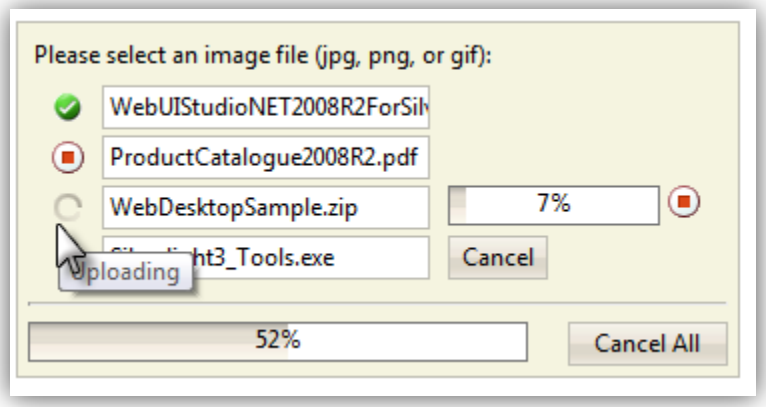
This whitepaper describes the concept and features introduced in WebFileUploader.

Table of Contents

Overview

Intersoft WebFileUploader is a memory-efficient ASP.NET file uploading component that supports asynchronous multiple files upload without page refresh. Unlike standard ASP.NET file uploader, WebFileUploader enables large file uploading with very minimum memory and resources consumption.

WebFileUploader introduces revolutionary experiences which enables you to select multiple files in a single instance and shows uploading progress in real time – making file uploading more intuitive, simpler and faster than ever. Furthermore, it also allows you to abort currently uploaded file, or cancel the pending uploads.



Intersoft WebFileUploader naturally integrates advanced uploading features with sophisticated user experiences, making file uploading more intuitive, faster and hassle-free.

Best of all, WebFileUploader uses 100% HTML technology for all its features, and fully supports all modern browsers including Firefox, Safari, Chrome and Opera. By 100% HTML technology, it means that WebFileUploader doesn't require Flash plug-in to support multiple files uploading and many other features that are previously only possible to be achieved with Flash plug-in – making WebFileUploader the most advanced, powerful and reliable file uploading component for the ASP.NET platform.

Concept

WebFileUploader is an easy to use; high performance and advanced file upload component which allows you to upload multiple files without page refresh.

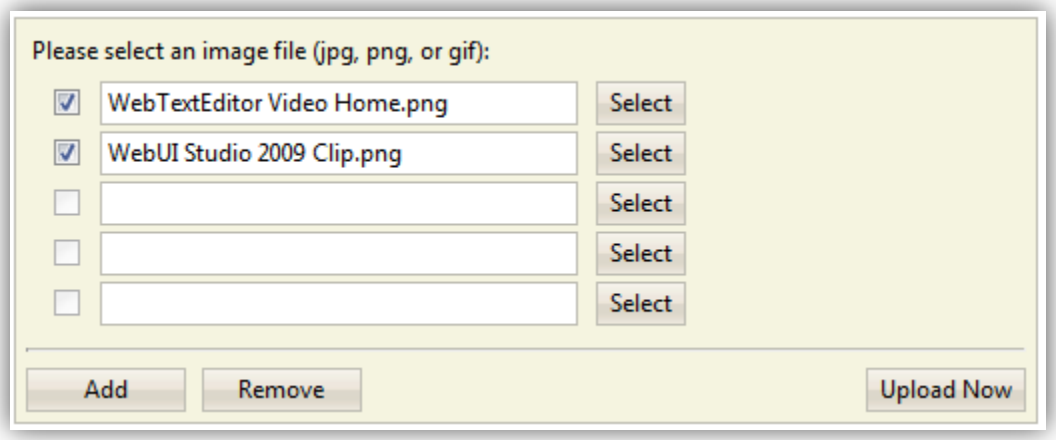
This section explains the fundamental concept of WebFileUploader to help you understand how it works, and to enable you leveraging many of its advanced features for your web application.

- **User Interface**

  WebFileUploader includes two user interface layout, panel and attachment bar.

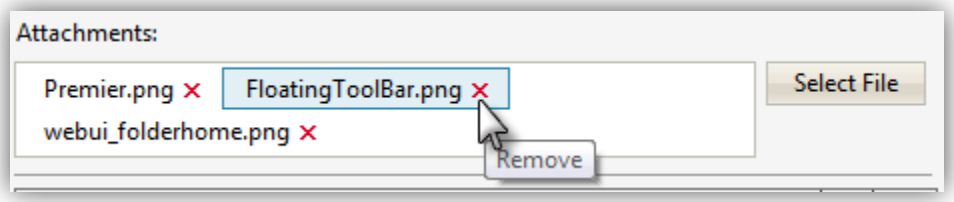
  Panel user interface layout provides rich user interface upload elements. There are checkbox, textbox, selection button, add button, remove button, upload button, cancel all button, upload

button. With panel layout, you can easily perform file selection, and then uploading multiple files, or cancelling running uploads.



Panel layout provides users with richer uploading experiences

AttachmentBar user interface layout is state-of-the-art design that minimizes the form of WebUploader component, making efficient use of screen real estate. It is specifically designed for attachment box in data entry form for uploading attachment files.



Perfect for data entry form with limited space, Attachment layout brings powerful multiple file uploading capabilities into a compact, minimal user interface.

- **Upload Mode**
  WebFileUploader comes with two uploading mode, automatic and batch upload.

  Automatic mode will automatically perform file uploading after a file is selected, while batch upload allows you to select several files and then upload them together in a single session.

  For batch upload mode, WebFileUploader provides customizable upload priority, such as priority by series, or by type.

- **User Interactions and Events**

  WebFileUploader is designed with robust architecture which allows various user interactions. Each interaction is properly recorded as status in the client object model, enabling you to easily inspect on the interaction status.

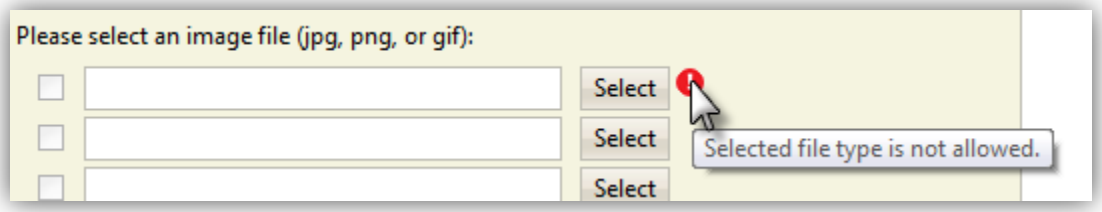
  WebFileUploader provides the following interactions:

  - File selecting
  - File selected
  - File added
  - File removed
  - File about to upload
  - File uploading
  - File uploaded
  - File canceled
  - File aborted

  In addition, WebFileUploader provides comprehensive client-side and server-side events which allow developers to perform additional processing or customization during the event. Some available events are before upload, uploading, after upload, and error event. Please refer to client-side and server-side reference for more details.

- **File Validation**

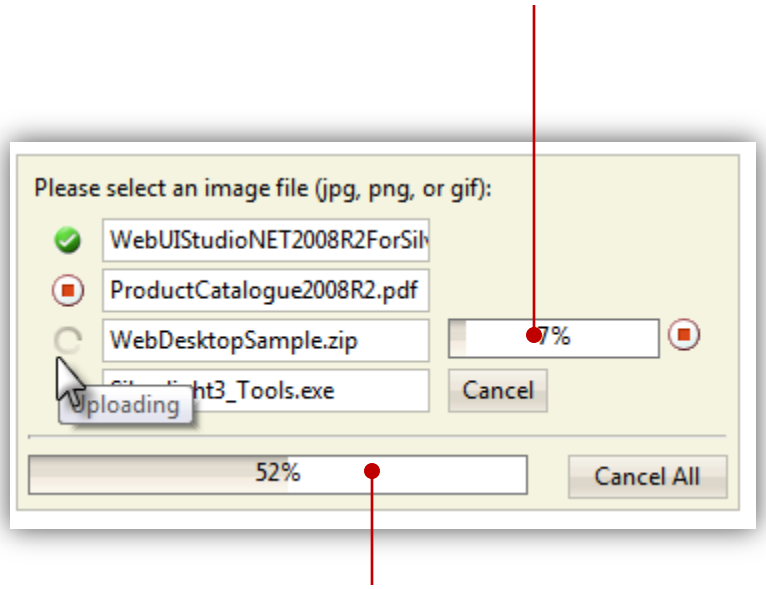
  WebFileUploader provides several built-in validations enabling you to easily limit files to be uploaded. You can validate files by type, redundant file, size and custom. In Panel layout, an error indicator will appear when invalid file is selected.



- **Progress Bar**

  One of the key features in WebFileUploader is its ability to show uploading progress in real-time. WebFileUploader provides rich user interface for progress bars. Each uploaded file has its own progress bar, and another total progress bar that shows uploading progress in overall.

The progress bar for currently uploaded file



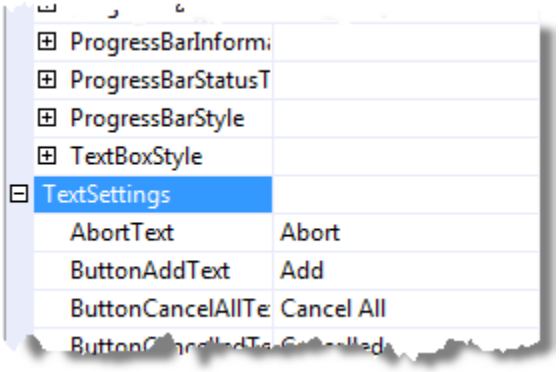
The overall progress bar shows the total completion percentage

Furthermore, you can show progress information by completion percentage, by size or by time remaining.

- **Customization**

  Customizing the look and appearance of WebFileUploader is easy and straightforward with skin and localization. WebFileUploader provides styling for user interface elements such as frame, textbox, button, text and more.

  Localization is also supported with text properties such as caption, tooltips, and texts and more. The text properties can be found in TextSettings object of the control.



WebFileUploader allows you to completely customize its appearance and textual settings

Features

WebFileUploader includes many advanced features to address the most demanding file uploading requirements such as discussed in the following.

- **Large files uploading with minimal resources consumption**

  The standard ASP.NET file upload component will often failed when you tried to upload large files, for instance, a 10 mb file. Two of the most common errors are request time out and out of memory error. That happened because the standard file uploading implementation in ASP.NET requires the file to be entirely uploaded before it returns a response to the client side.

  WebFileUploader takes advantage of HttpModule technology to handle large file upload with a more sophisticated implementation that doesn't perform memory-level buffering. As the result, WebFileUploader can handle large file uploads with very minimal memory and resources consumption.

  By default, ASP.NET application restricts maximum request length to 4MB. To enable your application to accept larger files, please configure the maxRequestLength in web.config to higher value. The maxRequestLength value is measured in kilobytes.

  The following configuration sets the application to allow file upload with size up to 100MB.

```
<configuration>
    <system.web>
        <httpRuntime maxRequestLength="102400" />
    </system.web>
</configuration>
```

  For IIS 7.0, there are additional settings that need to be configured.

  1. Set **maxAllowedContentLength** to a higher value, which is measured in bytes

     Open your application's web.config file, and add the following.

```
<system.webServer>
    <security>
      <requestFiltering>
              <requestLimits maxAllowedContentLength="102400000" />
      </requestFiltering>
    </security>
</system.webServer>
```

  2. Unlock the mode override in applicationHost.config

     Open IIS 7 application configuration file which is usually located in C:\windows\system32\inetsrv\config, and change the **overrideModeDefault** to **Allow**, such as shown in the following.

```
<section name="requestFiltering" overrideModeDefault="Allow" />
```

- **IIS 6 and IIS 7 Integrated mode support**

  WebFileUploader is built upon http module to provide reliable, memory-efficient implementation for large file uploads support. In addition, it also relies on http handler to support many advanced features such ability to abort currently uploaded files, cancel other pending uploads, and more.

  To use WebFileUploader in ASP.NET application that target IIS 6, please add the following entries to your web.config.

  - Add WebFileUploader hander to <httpHanders> section:

    ```
    <add verb="GET" path="WebFileUploaderHttpHandler.axd"
    validate="false"
    type="ISNet.WebUI.WebTextEditor.WebFileUploaderHttpHandler,
    ISNet.WebUI.WebTextEditor" />
    ```

  - Add WebFileUploader module to <httpModules> section:

    ```
    <add name="WebFileUploaderHttpModule"
    type="ISNet.WebUI.WebTextEditor.WebFileUploaderHttpModule,
    ISNet.WebUI.WebTextEditor">
    ```

  The following configuration applies for ASP.NET application that target IIS 7.

  - Add WebFileUploader hander to <handlers> section under <system.webServer>:

    ```
    <add name="WebFileUploaderHttpHandler" verb="GET"
    path="WebFileUploaderHttpHandler.axd"
    type="ISNet.WebUI.WebTextEditor.WebFileUploaderHttpHandler,
    ISNet.WebUI.WebTextEditor" preCondition="integratedMode" />
    ```

  - Add WebFileUploader module to <modules> section under <system.webServer>:

    ```
    <add name="WebFileUploaderHttpModule" preCondition="managedHandler"
    type="ISNet.WebUI.WebTextEditor.WebFileUploaderHttpModule,
    ISNet.WebUI.WebTextEditor">
    ```

- **Webfarm and multiple worker requests support with built-in FileStateServer**

  Due to AJAX uploading mechanism, most file upload components will fail to work when the application is used in webfarm environment, or in server running multiple worker requests.

  When running your application in such configuration, the load balancer automatically redirects the AJAX request to one of the available servers which may not be the originating server that receives the upload data. This mechanism affects the file upload component since the AJAX request may receive invalid response and thus unable to obtain the required upload information which finally failing the progress bar.

  Intersoft WebFileUploader addressed this limitation elegantly with its high-performance, reliable FileStateServer technology. When using FileStateServer, WebFileUploader streamlines all required uploading information in its state server, making it possible for AJAX request to receive real-time synchronized information regardless of the server that process the request.

  To enable FileStateServer in WebFileUploader to support webfarm configuration, simply add the following entries to your web.config:

```
<add key="ISNet.WebUI.WebFileUploader.ContextStorage"
      value="FileServer" />

<add key="ISNet.WebUI.WebFileUploader.ContextStorageFileServer"
      value="path=\\Server\ContextUploadStorage" />
```

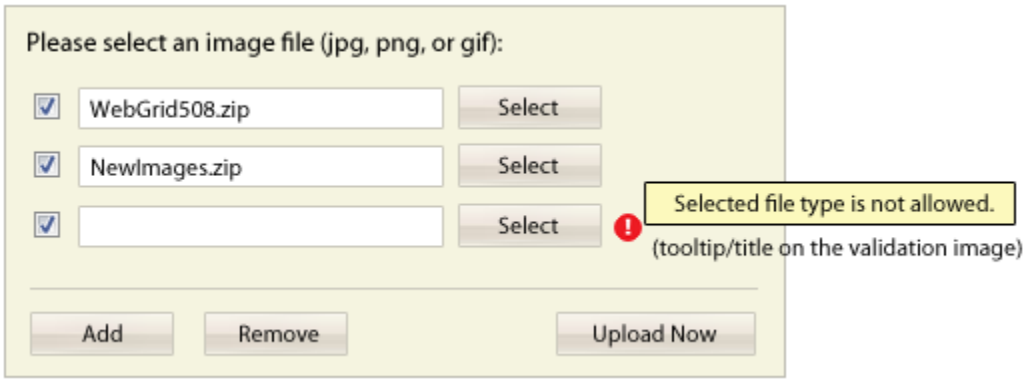
  Note: Make sure all your load balanced servers have the permissions and sufficient security settings to access the server and the shared resource. You may need to provide network credentials in your web.config to allow your application to access the specified server resource.

- **Rich user experiences**
  Sophisticatedly-designed interfaces and state-of-the-art experiences are what set WebFileUploader apart from other similar component in the industry. It provides end-to-end user experiences which makes file uploading easier and more intuitive than ever.

  WebFileUploader comes with two user interface for file uploading, panel layout and attachment bar layout.

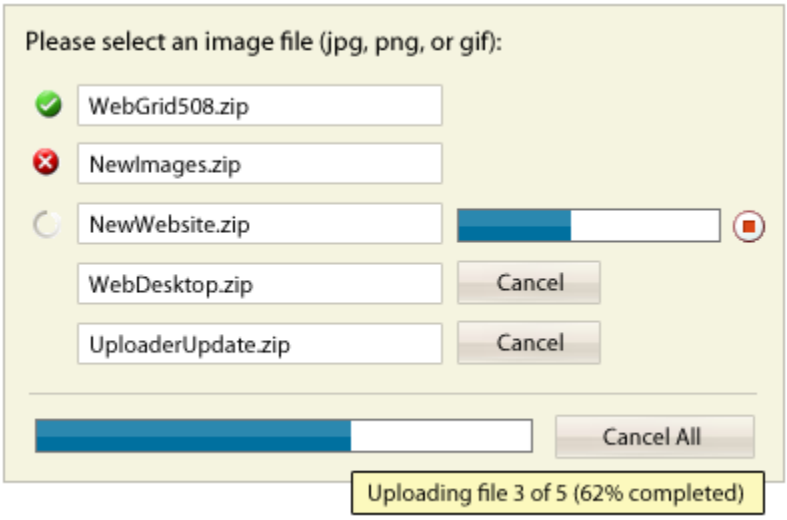
  The panel layout provides more comprehensive visual elements, and therefore is more suitable for uploading form. The panel layout offers streamlined user experiences such as explained below.

- **Selection Experience**



  - Upload visual elements are arranged elegantly, makes it intuitive to users.
  - When a validation error occurred, an error indicator is shown next to the Select button, makes it easy for users to notice the validation error. When hovered, the error message will be shown in the tooltip.

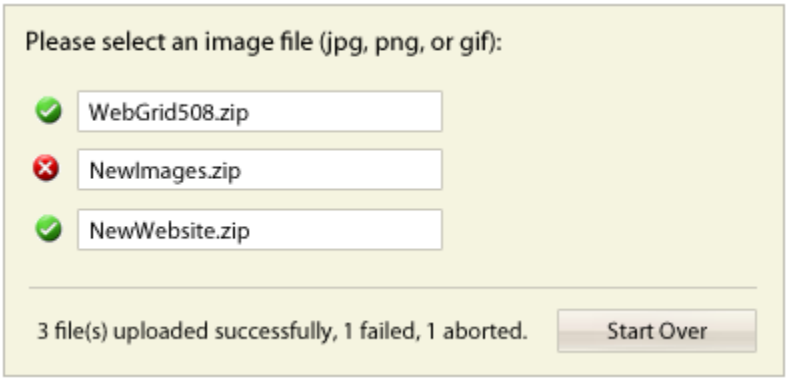
- **Uploading Experience**



  - When a file upload is in progress, a progress bar will be shown next to it displaying the current upload progress.
  - Intuitive status indicators make it easy for users to understand the file upload status. Successfully uploaded files will be marked with success indicator, as well as failed or canceled files. To see the error message of failed upload, simply mouse over on the error indicator.
  - User can abort the currently uploaded file by clicking on the Abort button next to the progress bar. When an upload is aborted, WebFileUploader will process to the next pending uploads.

- User can also cancel pending uploads by clicking on the Cancel button. Ultimately, user can click on Cancel All button in the status area to cancel all uploads. When cancel all is invoked, WebFileUploader will delete the successfully uploaded files, abort the current file in progress, and cancel pending uploads.
- The overall progress bar is shown in the status area displaying the total progress of all uploads. Hover on the overall progress bar area to see the detailed progress information.

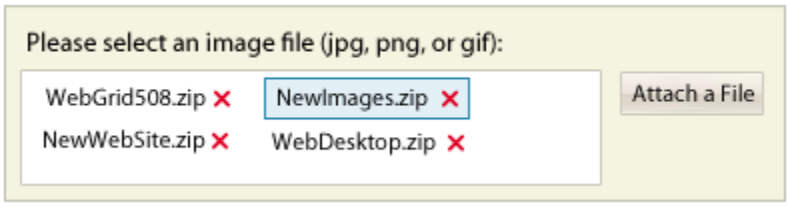
- **Completion Experience**



- When the upload session is completed, WebFileUploader sets the user interface to summary state, where the status area now displays the summary of the file uploads.
- The Start Over button is made visible in this state. When clicked, WebFileUploader will reset its state back to Selection Experience, making the file uploader ready to be used for the next upload session.

Similar to the experiences in Panel layout mode above, Attachment layout mode also provides streamlined user experience that makes uploading easy even in compact user interface such as explained below.

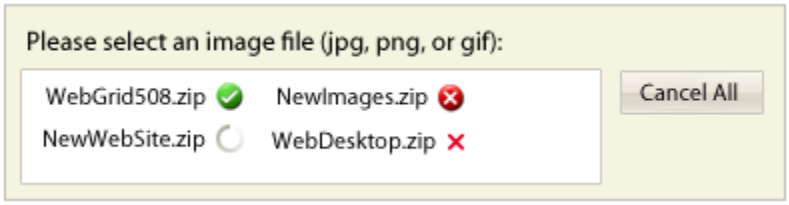
- **Selection Experience**



- Powerful uploading capabilities packed in minimal, fluid user interfaces.
- The attachment layout has only a button to attach a file. When selected, the file will be added to the main fluid container. The container will grow or shrink automatically as you add or remove files.
- To remove a file, simply click on the red X button, or select the file with a mouse click then press Delete button.

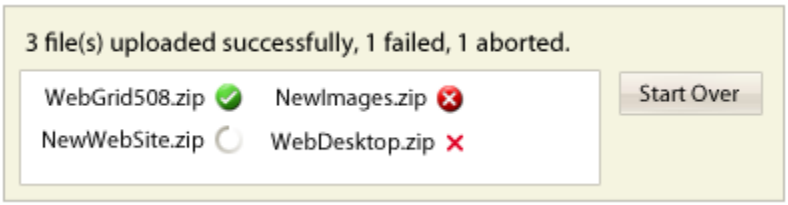
- The attachment layout doesn't include the command to perform uploading process. This is by design due to the nature of the user interface. In most cases, you may already have a button to perform data saving, where you can call WebFileUploader's client-side API to programmatically perform file uploading.

- **Uploading Experience**



  - WebFileUploader makes efficient use of screen real estate smartly by reusing the space of unnecessary existing visual elements according to the upload context.
  - Status indicators will be shown next to the file. Simply hover on it to see its detailed status information.
  - To cancel pending uploads, simply click on the red X button. User can also cancel all uploads by clicking on the Cancel All button.

- **Completion Experience**



  - When all files have been processed, WebFileUploader sets the user interface to summary state. Notice that the upload summary is now displayed in the caption bar.
  - Click on Start Over to go back into Selection Experience for the next upload session.

- **Built-in file saving**

  Unlike standard ASP.NET file upload, WebFileUploader doesn't require you to write codes to save uploaded files. Just set the **UploadPath** property of the WebFileUploader control and you're all set.

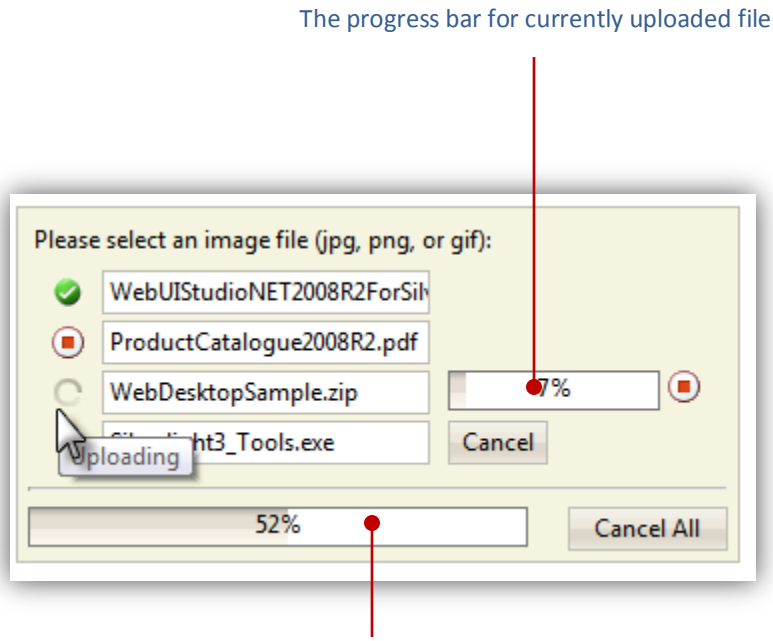
  WebFileUploader automatically saves all uploaded files to the specified upload path. Make sure you have configured sufficient security and permission settings for ASP.NET worker process to acess the specified upload path.

  You can enter physical path, IIS virtual path, or network path for the upload location. The default upload path is ~/Upload.

- **Real-time progress bar**

  Intersoft WebFileUploader revolutionizes uploading experience with innovative, real-time progress bar feature. With this feature, user can now see the current uploading progress directly within the file upload interface, making file uploading more engaging and intuitive. Furthermore, WebFileUploader also provides more sophisticated progress information such as uploaded size and estimated completion time.

  WebFileUploader shows two types of progress bar, for each file upload and for overall upload progress status.



  In addition, WebFileUploader provides comprehensive progress bar options enabling you to customize the progress bar behaviors. For instances, you can disable the overall progress bar, or change the progress information type to show uploaded size instead of percent completed.

  The following is a list of properties for customizing progress bar:

  - **ProgressLatency**. This property lets you specify the time latency for progress bar update. Default value is 500 milliseconds.
  - **ProgressInformationType**. The information type for individual upload progress. Default value is Percentage, while other options are Size and TimeRemaining.
  - **ProgressInformationTypeForTotal**. Similar to ProgressInformationType, this property is to set the information type for the overall progress bar.
  - **ShowProgressArea**. Determines whether progress bar area should be shown.

- **ShowProgressBar**. Determines whether the progress bar for each upload should be shown.
- **ShowProgressBarForTotal**. Determines whether the progress bar for overall upload should be shown.

WebFileUploader also provides comprehensive styles for customizing the appearance of the progress bar. You can customize the progress bar frame, the information text, the status text, the mask style and more.

Note: The progress bar feature requires additional http handler registration on your web.config. Please refer to IIS Support for more details.

- **Automatic upload**

  With automatic upload feature, each selected file will be uploaded immediately without any additional efforts. This feature is suitable for data form that accepts only a single file upload. However, it could be useful in some other advanced scenarios as well, such as in mail application.

  To use automatic upload behavior, simply set **UploadType** of the uploader control to Automatic value.

- **Batch upload**

  Batch upload is the default upload type in WebFileUploader. With batch upload, users can add multiple files in the uploader interface, and then upload all selected files in a single session.

  Batch upload provides more intuitive and better controls on the uploading process – such as users can add several files, or remove unnecessary files or change the files – making it the preferred upload type for most file uploading scenarios.

- **Batch upload with priority**

  When using batch upload type, you can also configure the upload priority which decides the logical order of the files to be uploaded.

  You can set the upload priorty by series, which is the default value, or by type. By series means the files will be uploaded in the original sequence as they were added in the uploader interface, while by type means files with same type will be uploaded first, then following the other types in alphabetical order.

To configure upload priority, simply set **PriorityUploadType** property.

- **Simultaneous batch upload**

  By default, WebFileUploader uploads one file at a time to avoid over consumption on bandwidth and resources. However, WebFileUploader can also be configured to upload multiple files asynchronously to support enterprise-level usage, such as internal or corporate web application.

  Since WebFileUploader will upload all files simultaneously, please use this feature with caution as it will significantly affect your server's workload and resources consumption. Make sure you have load-tested your server properly to use this feature.

  You may also want to enable validations on the total upload size or allowed file types for security purpose.

  To enable simultaneous uploads, just set *AllowSimultaneousUpload* to true.

- **Client-side events**

  WebFileUploader is designed with rock-solid API and extensible client-side events to give you total control over uploading process. Works in conjunction with real-time progress bar feature, WebFileUploader sent event notification to client-side in real-time – such as OnBeforeUpload, OnAfterUpload OnErrorUpload, and even OnUploading event.

  The following lists the available client side events in more details:

  - **OnInitialize**. Fired once when the control is initialized.
  - **OnBeforeUpload**. Fired when a file is about to be uploaded.
  - **OnAfterUpload**. Fired when a file has been successfully uploaded.
  - **OnUploading**. Fired when uploader control receives progress information on a file being uploaded.
  - **OnCancelUpload**. Fired when a file upload is cancelled.
  - **OnErrorUpload**. Fired when a file upload is failed due to error.

  The following example shows how to handle the client-side events, and shows the information passed to the parameter of each event.

```
<ISWebTextEditor:WebFileUploader ID="WebFileUploader1" runat="server"
      Width="450px" UploadPath="./Upload" AllowAdd="false"
      AllowRemove="false">
   <ClientSideEvents OnAfterUpload="doAfterUpload"
       OnBeforeUpload="doBeforeUpload"OnCancelUpload="doCancelUpload"
       OnInitialize="doInitialize" OnUploading="doProcessing"
       OnError="doErrorUpload" />
</ISWebTextEditor:WebFileUploader>
```

```
function doAfterUpload(id, webFileUploadInfo)
{
    WriteLog("AfterUpload Event : File " +
          webFileUploadInfo.GetFileName() + " is uploaded.");
}

function doErrorUpload(id, webFileUploadInfo, errorMessage,
    errorStackTrace)
{
    WriteLog("ErrorUpload Event : File " +
          webFileUploadInfo.GetFileName() + " is failed to upload due
          to error '" + errorMessage + "'");
}

function doBeforeUpload(id, webFileUploadInfo)
{
    WriteLog("BeforeUpload Event : File " +
          webFileUploadInfo.GetFileName() + " is about to upload.");
}

function doCancelUpload(id, webFileUploadInfo)
{
    WriteLog("CancelUpload Event : File " +
          webFileUploadInfo.GetFileName() + " is cancelled.");
}

function doInitialize(id)
{
    WriteLog("Initialize Event : Initialize " + id + " control.");
}

function doProcessing(id, webFileUploadInfo)
{
    WriteLog("Uploading Event : Uploading file " +
          webFileUploadInfo.GetFileName() + " is " +
          webFileUploadInfo.UploadedLength + " bytes of " +
          webFileUploadInfo.ContentLength + " bytes.");
}
```

- **Server-side events**

  In addition to client-side events, WebFileUploader also provides comprehensive server-side events, making it easy for developers to customize and extend the control with additional processing on certain event.

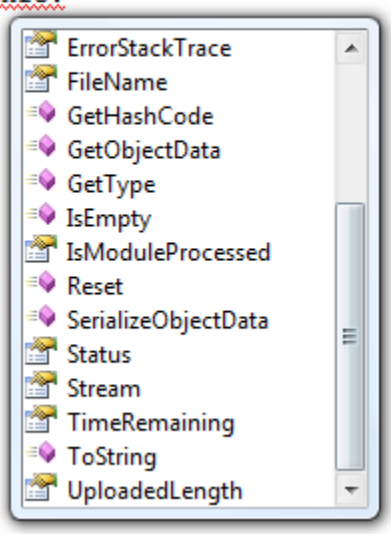
  WebFileUploader is designed with sophisticated architecture that enables server-side events to be raised properly, although WebFileUploader doesn't have direct access to the Page object due to the nature of its architecture that used more advanced HttpModule/HttpHandler technology instead of old-fashioned AJAX.

  The following lists server-side events available in WebFileUploader.

- **Uploading**. Fired when a file upload is in progress. This event will be invoked continuously during the upload progress.
- **AfterUpload**. Fired when a file upload is succeeded.
- **CancelUpload**. Fired when a file upload is cancelled.
- **ErrorUpload**. Fired when a file upload is failed due to error.

All server-side events receive **WebFileUploaderFileEventArgs** parameter that contains information about an upload file in the context.

```
protected void WebFileUploader1_AfterUpload(ob
{
    e.WebFileUploadInfo.
}
```



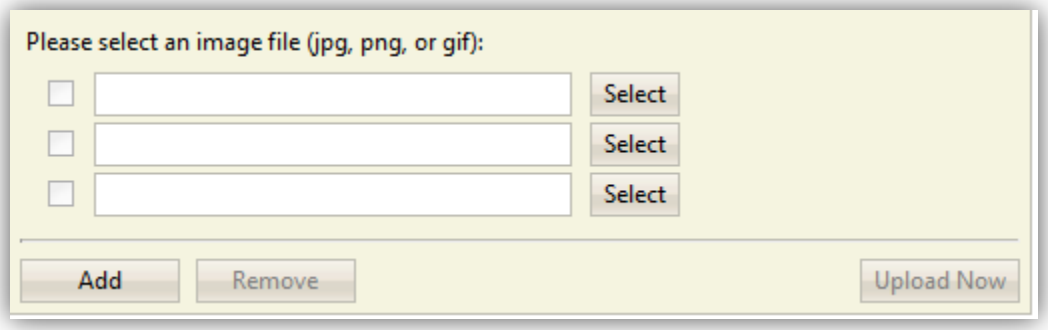
The WebFileUploadInfo object provides complete information about an upload file, such as file name, status, error message and more.

The following C# example shows how to move the uploaded file to another folder after a successful file upload.

```
protected void WebFileUploader1_AfterUpload(object sender,
      WebFileUploaderFileEventArgs e)
{
   try
   {
      if (File.Exists(e.WebFileUploadInfo.FileName))
          File.Move(e.WebFileUploadInfo.FileName,
                  Path.Combine(userFolder,
                  Path.GetFileName(e.WebFileUploadInfo.FileName)));
   }
   catch (Exception e)
   {
      WriteErrorLog("Unable to move uploaded file. Error details: " +
            e.StackTrace);
   }
}
```

- **Initial files count**

  WebFileUploader gives you the flexibility to set the number of initial file upload fields that displayed in panel user interface. This property is not applicable for attachment bar user interface.



WebFileUploader is configured to show only three upload fields.

  To customize this setting, simply set the **InitialFileCount** property to a number that you desire.

- **Limit upload files by type, count, or custom**

  WebFileUploader includes sophisticated built-in validation before user can select the file to be uploaded. It can perform validation by file types, file count, and custom validation.

  By default, WebFileUploader automatically validates redundant files. That means you can't select the same file that already been added.

  To limit file types, set the **FileTypes** property to the file extensions with mask. You can enter multiple file types by separating each type with a semicolon separator, such as *.png;*.gif;*.jpg

  To limit files count, simply set the **FilesCount** to an integer number.

  Note: The file types, file count and custom validation is performed in client-side, while file size validation is performed in server-side.

  Code Example:

```
<ISWebTextEditor:WebFileUploader ID="WebFileUploader1" runat="server"
        FilesCount="5" FileTypes="*.zip;*.rar" Width="500px">
</ISWebTextEditor:WebFileUploader>
```

- **Limit upload size and total upload size**

  In most cases, your application would need to limit the upload size of an individual file for security and resource management purpose. By default, WebUploader doesn't set upload size limit.

  To customize upload size limit, please set **UploadSizeLimit** property to an integer value with the scale of byte.

  In addition, WebFileUploader also enables you to limit the total size of all uploaded files. When it reach the maximum size, it will abort the current file being uploaded and cancel the rest pending uploads.

  To configure total upload size limit, please set **TotalUploadSizeLimit** property to an integer value with the scale of byte.

  The following example shows a WebFileUploader with individual upload size configured to 10 kilobytes and total upload size to 100 kilobytes.

```
<ISWebTextEditor:WebFileUploader ID="WebFileUploader1" runat="server"
         TotalUploadSizeLimit="100000" UploadSizeLimit="10000">
</ISWebTextEditor:WebFileUploader>
```