

Working with File Uploads

Adding a file upload feature to your web site is relatively simple, but there are some basic questions you should answer first before writing code. The first question you want to ask is what types of files are allowed. For example - if your intent is to allow someone to upload a resume, most likely you want to allow for Word docs, PDFs, and TXT files. If you site is letting someone upload a picture, than you want to restrict the types to GIF, JPG and PNG. You may also want to go further than just checking the file extension. You may want to use a third party tool like Alagad Image CFC or ColdFusion 8's built in image support to not only confirm that the file is indeed an image, but also ensure that the dimensions aren't too big.

The next consideration to make is where these files will be stored. You have a few concerns here:

1) The first concern is a basic one - disk space. If you are letting people upload images, it is possible that your hard drive could get filled. With the sizes of today's hard drives and with most of us not getting a lot of traffic (comparatively), you may never even get close to filling your hard drive. The question is though - how would you know? There is no built in way within ColdFusion to determine the amount of free space available on a drive (although CFLib has two different Windows only options, [AvailableSpace](http://www.cflib.org/udf.cfm?ID=101) and [FreeSpace](http://www.cflib.org/udf.cfm?ID=105)). You could however simply generate a report of the total size of one folder. This can be done using a recursive cfdirectory and query of query:

```
<cfdirectory directory="#expandPath('.')#" recurse="true" action="list" name="allfiles">
<cfquery name="getSize" dbtype="query">
select  sum(size) as total
from    allfiles
</cfquery>
<cfoutput>Total size in bytes is: #getSize.total#</cfoutput>
```

This could be done daily and sent via email using a scheduled task. Or it could be displayed in the your web site's administration section. Either way - you want to at least have a good idea of how much space is being used and how much space is available. If your site gets a lot of traffic and could possibly have quite a few uploads, you may need to look into using another server just for the storage and serving of the media files.

2) The next question is how you will handle files with the same name. ColdFusion provides a simple way to handle that when uploading. It has a makeUnique feature that will automatically detect a file name conflict and give your upload a new name, but you need to be sure you use this (if it makes sense) and that you recognize the file changed. If you store the file name in the database (a good idea so that you can associate uploads with a user) you may want to consider storing both the original file and the file name ColdFusion used. This lets the user see his original (and recognized) file name while keeping a link to the real file name on the system.

So hopefully now at this point you have an idea of what you want to do with the files - now lets look at the actual code. First and foremost you will need a form. The only way a browser can upload a file is via a form that uses a `<input>` tag with type of file. There is no other way (outside of a Java or Flash file) that you can let users upload a file. So a very, very simple upload form could look like so:

```
<form action="test.cfm" method="post">
Select a file: <input type="file" name="uploadfile">
<input type="submit" value="Upload File">
</form>
```

While this looks ok, there is one problem. When using a file upload, you have to specify a special type of form. You won't see a problem until you actually try to upload the file with CFFILE. The error you will see is:

The cffile action="upload" requires forms to use enctype="multipart/form-data"

Well you have to hand it to ColdFusion. It told you exactly what was wrong, and don't feel too bad - I forget this all the time. All you need to do is add the specified enctype attribute to your form tag, like so:

```
<form action="test.cfm" method="post" enctype="multipart/form-data">
Select a file: <input type="file" name="uploadfile">
<input type="submit" value="Upload File">
</form>
```

So this fixes the form - now lets move on to the processing. As hinted at above - the CFFILE tag will actually do the upload. To be clear - the web server handles the file upload, but the CFFILE tag is what actually lets your process the upload. It gives you a way to say, "Take the user's file and put it here." Let's look at a simple example:

```
<cfif structKeyExists(form, "uploadfile")>
  <cfset destination = expandPath("./files")>
  <cfif not directoryExists(destination)>
    <cfdirectory action="create" directory="#destination#">
  </cfif>

  <cffile action="upload" filefield="uploadfile" destination="#destination#"
nameConflict="makeUnique" result="upload">
  <cfdump var="#upload#">
</cfif>
```

This code block will only run when the form field, uploadfile, exists. The first 4 lines simply check to see if a subdirectory named files exists. Normally you would probably use a folder that already exists, but this is a handy way to set it up for me. (I'm a lazy SOB.) The real critical line here is the CFFILE tag. Let's look at each attribute. The first attribute, action, specifies that we are doing an upload. (Duh.) The next attribute is interesting. You tell ColdFusion the name of the form field that contains the file. Notice I didn't use #s or form. I just used the form field name. (And yes - you can have more than one file upload in a form.) Next I specify a destination folder, using the variable I created earlier.

Now pay special attention to the nameConflict attribute. As I mentioned above, if you have multiple users uploading files (or even just one user uploading multiple files), the chance that two files could have the same name will grow. You have 4 options for this attribute:

- Error: If error is used and a file name conflicts, an error will be thrown.
- Skip: The file will simply not be saved.
- Overwrite: This will overwrite the existing file.
- MakeUnique: This will create a new file name.

The last attribute, result, simply specifies that the result of the action should be stored in a structure named upload. In ColdFusion 6 and earlier, this variable was always named CFFILE. So what gets stored in the structure? Quite a bit actually. Here are the keys you can expect to be returned:

- attemptedserverfile: This is the file name ColdFusion tries to use for the file it will save. As far as I can see, this seems to match with the name of file on your system. Most likely the name would only change if you uploaded a file with a name that was valid on your system and not valid on ColdFusion's host operating system.
- clientDirectory: The directory of the file that user upload.
- clientFile: The file the user uploaded.
- clientFileExt: The extension of the file the user upload.
- clientFileName: This is a weird one. This value is the filename of the file uploaded - minus the extension. I get confused by this all the time.
- contentType and contentSubType: The MIME content type and subtype of the file uploaded. This can be used to restrict uploads to certain types of files.
- dateLastAccessed: The docs say this is the date and time the file was last accessed. When I look

at this result though I see only the date.

- `fileExisted`: This is a boolean that tells you if a file with the same name existed when you ran the `CFFILE` tag. If you use `nameConflict="skip"`, this would be one way to determine if a skip action occurred.
- `fileSize`: The size of the uploaded file.
- `fileWasAppended`: A boolean that tells you if the file was appended to an existing file. Since "append" isn't even a valid option, I'm not quite sure why this exists.
- `fileWasOverwritten`: A boolean that tells you if the file was overwritten. This would only work if `nameConflict` was set to `overwrite`. Otherwise it will always be false.
- `fileWasRenamed`: A boolean that tells you if the file was renamed. This would only work if `nameConflict` was set to `makeUnique`. Otherwise it will always be false.
- `fileWasSaved`: A boolean that tells you if the file operation completed. This would only be false if `skip` was used for `nameConflict`.
- `oldFileSize`: If a file upload overwrites another file, this will tell you the size of the old file. Not quite sure how useful this is.
- `serverDirectory`: The directory where the file was saved.
- `serverFile`: The filename of the file. Useful when `makeUnique` is used.
- `serverFileExt`: The extension of the uploaded file.
- `serverFileName`: Like `clientFileName`, this is the filename of the file minus the extension.
- `timeCreated` and `timeLastModified`: The create and lastmodified values for the uploaded file. Unlike `dateLastAccessed`, this seems to be a complete date and time stamp.

Ok, so a lot was covered there. Basically the structure gives you everything you need to handle the post-upload process. Need to store the uploaded file name and remember the original file name? The data is there. Want to use ColdFusion 8's image functions to ensure the image is the right size? The data is there. (And if you want an example of that - download the sample code from my blog post on `CFIMAGE`: [Recording from ColdFusion 8 Image Presentation](http://www.coldfusionjedi.com/index.cfm/2007/8/9/Recording-from-ColdFusion-8-Image-Presentation).)

Security Concerns

Before allowing a user to upload a file, think long and hard about what this means for security. First off - will you be storing the files under web root? If so - it means anyone can then link to the file - whether or not you intended them to do so. You may have a nice, family oriented web site, but if you let folks upload images, there is a good chance someone could upload porn, and then link to it from other sites. Most likely you do not want to store any uploaded file under the web root. You also want to make it easy to review uploaded files. You could fire off an email everytime someone uploads a file. You could also build an admin interface that lets you browse uploaded files and double check them for appropriateness.

Another concern would be allowing users to upload code. Imagine if you let users upload anything and they decided to upload a CFM file? The user would then be able to execute any code against your server they want if the file was stored under web root. You definitely want to prevent this by ensuring that the file isn't under web root.

One last tip you may want to consider. If you do want to limit what types of files can be uploaded, consider the `accept` attribute. This attribute takes a list of MIME types you want to allow. The docs demonstrate allowing Word and JPG files like so: `accept = "image/jpeg,application/msword"`. Two small issues with this. First - you need to know the MIME types. Since my brain only has room for useless Star Wars trivia, I usually have to look this up. (Here is a good URL for a list of MIME types: http://www.w3schools.com/media/media_mimeref.asp) The second thing to consider is that the `accept` attribute only checks the extension. A user could write a text file, save it with a `.jpg` extension, and the `accept` attribute would gladly let it pass. Note that ColdFusion 8 provides native functions to see if a file is a PDF or image. These can be used along with `accept` to really ensure a file is proper.