php

- Downloads
- Documentation
- Get Involved
- Help

Search

PHP 5.4.31 Released

Keyboard Shortcuts

?	This help
j	Next menu item
k	

Previous menu item

g p

Previous man page

g n

Next man page

G

Scroll to bottom

g g

Scroll to top

g h

Goto homepage

g s

Goto search
(current page)

/

Focus search box

POST method uploads »
« Dealing with XForms

- PHP Manual
- Features

Change language: English

Edit Report a Bug

Handling file uploads ¶

Table of Contents ¶

add a note

User Contributed Notes 38 notes

up
down
75
CertaiN ¶
6 months ago

```
You'd better check $_FILES structure and values throughly.
```

```
The following code cannot cause any errors absolutely.

Example:
<?php

header('Content-Type: text/plain; charset=utf-8');

try {

    // Undefined | Multiple Files | $_FILES Corruption Attack
    // If this request falls under any of them, treat it invalid.
    if (
        !isset($_FILES['upfile']['error']) ||
        is_array($_FILES['upfile']['error'])
    ) {
        throw new RuntimeException('Invalid parameters.');
    }

    // Check $_FILES['upfile']['error'] value.
    switch ($_FILES['upfile']['error']) {
        case UPLOAD_ERR_OK:
            break;
        case UPLOAD_ERR_NO_FILE:
            throw new RuntimeException('No file sent.');
        case UPLOAD_ERR_INI_SIZE:
        case UPLOAD_ERR_FORM_SIZE:
            throw new RuntimeException('Exceeded filesize limit.');
        default:
            throw new RuntimeException('Unknown errors.');
    }

    // You should also check filesize here.
    if ($_FILES['upfile']['size'] > 1000000) {
        throw new RuntimeException('Exceeded filesize limit.');
    }

    // DO NOT TRUST $_FILES['upfile']['mime'] VALUE !!
    // Check MIME Type by yourself.
    $finfo = new finfo(FILEINFO_MIME_TYPE);
    if (false === $ext = array_search(
        $finfo->file($_FILES['upfile']['tmp_name']),
        array(
            'jpg' => 'image/jpeg',
            'png' => 'image/png',
            'gif' => 'image/gif',
        ),
        true
    )) {
        throw new RuntimeException('Invalid file format.');
    }

    // You should name it uniquely.
    // DO NOT USE $_FILES['upfile']['name'] WITHOUT ANY VALIDATION !!
    // On this example, obtain safe unique name from its binary data.
    if (!move_uploaded_file(
        $_FILES['upfile']['tmp_name'],
        sprintf('./uploads/%s.%s',
            sha1_file($_FILES['upfile']['tmp_name']),
            $ext
        )
    )) {
        throw new RuntimeException('Failed to move uploaded file.');
    }

    echo 'File is uploaded successfully.';

} catch (RuntimeException $e) {

    echo $e->getMessage();

}

?>
```

up
down
15
keith at phpdiary dot org ¶
9 years ago

```
Caution: *DO NOT* trust $_FILES['userfile']['type'] to verify the uploaded filetype; if
you do so your server could be compromised.  I'll show you why below:

The manual (if you scroll above) states: $_FILES['userfile']['type'] -  The mime type of
the file, if the browser provided this information. An example would be "image/gif".

Be reminded that this mime type can easily be faked as PHP doesn't go very far in
verifying whether it really is what the end user reported!

So, someone could upload a nasty .php script as an "image/gif" and execute the url to the
"image".

My best bet would be for you to check the extension of the file and using exif_imagetype()
to check for valid images.  Many people have suggested the use of getimagesize() which
returns an array if the file is indeed an image and false otherwise, but exif_imagetype()
is much faster. (the manual says it so)
```

up
down

9

info at levaravel dot com ¶

5 years ago

```
A little codesnippet which returns a filesize in a more legible format.

<?php

function display_filesize($filesize){

    if(is_numeric($filesize)){
    $decr = 1024; $step = 0;
    $prefix = array('Byte','KB','MB','GB','TB','PB');

    while(($filesize / $decr) > 0.9){
        $filesize = $filesize / $decr;
        $step++;
    }
    return round($filesize,2).' '.$prefix[$step];
    } else {

    return 'NaN';
    }

}

?>
```

up

down

4

jan at lanteraudio dot nl ¶

1 year ago

```
Also stumbled on the max_file_size problem, in particular getting no response, no error
whatsoever when uploading a file bigger than the set upload_max_filesize.

I found that it's not the upload_max_filesize setting, but instead the post_max_size
setting causing this no response issue. So if you set post_max_size way larger than
upload_max_filesize, at least you are likely to get an error response when filesize
exceeds upload_max_filesize but is still within the limits of post_max_size.

Hope this helps anyone.
```

up

down

4

Thomas ¶

2 years ago

```
MIME type can be faked.

VVV
```

$_FILES['userfile']['type']

The mime type of the file, if the browser provided this information. An example would be "image/gif". This mime type is however not checked on the PHP side and therefore don't take its value for granted.

http://www.php.net/manual/en/features.file-upload.post-method.php

[Editor's note: removed a reference to a deleted note, and edited the note to make sense by itself.]

up
down
7
svenr at selfhtml dot org ¶
7 years ago

Clarification on the MAX_FILE_SIZE hidden form field:

PHP has the somewhat strange feature of checking multiple "maximum file sizes".

The two widely known limits are the php.ini settings "post_max_size" and "upload_max_size", which in combination impose a hard limit on the maximum amount of data that can be received.

In addition to this PHP somehow got implemented a soft limit feature. It checks the existance of a form field names "max_file_size" (upper case is also OK), which should contain an integer with the maximum number of bytes allowed. If the uploaded file is bigger than the integer in this field, PHP disallows this upload and presents an error code in the $_FILES-Array.

The PHP documentation also makes (or made - see bug #40387 - http://bugs.php.net/bug.php?id=40387) vague references to "allows browsers to check the file size before uploading". This, however, is not true and has never been. Up til today there has never been a RFC proposing the usage of such named form field, nor has there been a browser actually checking its existance or content, or preventing anything. The PHP documentation implies that a browser may alert the user that his upload is too big - this is simply wrong.

Please note that using this PHP feature is not a good idea. A form field can easily be changed by the client. If you have to check the size of a file, do it conventionally within your script, using a script-defined integer, not an arbitrary number you got from the HTTP client (which always must be mistrusted from a security standpoint).

up
down
2
warwickbarnes at yahoo dot co dot uk ¶
8 years ago

You may come across the following problem using PHP on Microsoft IIS: getting permission

```
denied errors from the move_uploaded_file function even when all the folder permissions
seem correct. I had to set the following to get it to work:

1. Write permissions on the the folder through the IIS management console.
2. Write permissions to IUSR_'server' in the folder's security settings.
3. Write permissions to "Domain Users" in the folder's security settings.

The third setting was required because my application itself lives in a secure folder -
using authentication (either Basic or Windows Integrated) to identify the users. When the
uploads happen IIS seems to be checking that these users have write access to the folder,
not just whether the web server (IUSR_'server') has access.

Also, remember to set "Execute Permissions" to "None" in the IIS management console, so
that people can't upload a script file and then run it. (Other checks of the uploaded file
are recommended as well but 'Execute None' is a good start.)
```

up
down
2
Tyfud ¶
9 years ago

```
It's important to note that when using the move_uploaded_file() command, that some
configurations (Especially IIS) will fail if you prefix the destination path with a
leading "/". Try the following:

<?php move_uploaded_file($tmpFileName,'uploads/'.$fileName); ?>

Setting up permissions is also a must. Make sure all accounts have write access to your
upload directory, and read access if you wish to view these files later. You might have to
chmod() the directory or file afterwards as well if you're still getting access errors.
```

up
down
1
ragtime at alice-dsl dot com ¶
6 years ago

```
I don't believe the myth that 'memory_size' should be the size of the uploaded file. The
files are definitely not kept in memory... instead uploaded chunks of 1MB each are stored
under /var/tmp and later on rebuild under /tmp before moving to the web/user space.

I'm running a linux-box with only 64MB RAM, setting the memory_limit to 16MB and uploading
files of sizes about 100MB is no problem at all! Nevertheless, some users reported a
problem at a few 100MB, but that's not confirmed... ;-)

The other sizes in php.ini are set to 1GB and the times to 300... maybe the execution_time
limits before, since the CPU is just a 233MHz one... :-)

====

OK,... I got it... finally!
```

```
If some of you have also problems uploading large files but the usual sizes/times in
php.ini are ok, please check

    session.gc_maxlifetime

when you are using session management with your upload script!

The default value is 1440 which is just 24min... so with only 600kbit/s upload rate the
session will be closed automatically after uploading
about 100MB. Actually you are able to upload more, but the file won't be copied from the
temporary to the destination folder... ;-)

You can set the value also directly inside the php-script via
<?php ini_set("session.gc_maxlifetime","10800"); ?>
```

up
down
3
robpet at tds dot net ¶
9 years ago

```
People have remarked that incorrect permissions on the upload directory may prevent photos
or other files from uploading.  Setting the Apache owner of the directory incorrectly will
also prevent files from uploading -- I use a PHP script that creates a directory (if it
doesn't exist already) before placing an uploaded file into it.  When the script creates
the directory and then copies the uploaded file into the directory there is no problem
because the owner of the file is whatever Apache is running as, typically "nobody".
However, lets say that I've moved the site to a new server and have copied over existing
file directories using FTP.  In this case the owner will have a different name from the
Apache owner and files will not upload. The solution is to TelNet into the site and reset
the owner to "nobody" or whatever Apache is running as using the CHOWN command.
```

up
down
1
Leevi at izilla dot com dot au ¶
9 years ago

```
This may help a newbie to file uploads.. it took advice from a friend to fix it..

If you are using
-windows xp
-iis 5
-php 5

If you keep getting permission errors on file uploads... and you have sworn you set the
permissions to write to the directory in iis...

double check that
a) in windows explorer under tools > folder options
click the view tab
```

```
scroll down all the way to "use simple file sharing (recommended)"
uncheck this box

b) find the folder you wish to upload to on your server
c) click properties and then the security tab
d) make sure the appropriate write settings are checked.

you may want to test by setting "everyone" to have full permission....

BEWARE doing this will open up big security holes on your server....

hope this helps

Leevi Graham
```

up
down
1
xmontero at dsitelecom dot com ¶
2 years ago

```
If "large files" (ie: 50 or 100 MB) fail, check this:

It may happen that your outgoing connection to the server is slow, and it may timeout not
the "execution time" but the "input time", which for example in our system defaulted to
60s. In our case a large upload could take 1 or 2 hours.

Additionally we had "session settings" that should be preserved after upload.

1) You might want review those ini entries:

* session.gc_maxlifetime
* max_input_time
* max_execution_time
* upload_max_filesize
* post_max_size

2) Still fails? Caution, not all are changeable from the script itself. ini_set() might
fail to override.

More info here:
http://www.php.net/manual/es/ini.list.php

You can see that the "upload_max_filesize", among others, is PHP_INI_PERDIR and not
PHP_INI_ALL. This invalidates to use ini_set():
http://www.php.net/manual/en/configuration.changes.modes.php

Use .htaccess instead.

3) Still fails?. Just make sure you enabled ".htaccess" to overwrite your php settings.
```

This is made in the apache file. You need at least AllowOverride Options.

See this here:
http://www.php.net/manual/en/configuration.changes.php

You will necessarily allow this manually in the case your master files come with AllowOverride None.

Conclussion:

Depending on the system, to allow "large file uploads" you must go up and up and up and touch your config necessarily up to the apache config.

Sample files:

These work for me, for 100MB uploads, lasting 2 hours:

In apache-virtual-host:

```
-----------------------------------------------------------
<Directory /var/www/MyProgram>
    AllowOverride Options
</Directory>
-----------------------------------------------------------
```

In .htaccess:

```
-----------------------------------------------------------
php_value session.gc_maxlifetime 10800
php_value max_input_time         10800
php_value max_execution_time     10800
php_value upload_max_filesize    110M
php_value post_max_size          120M
-----------------------------------------------------------
```

In the example,
- As I last 1 to 2 hours, I allow 3 hours (3600x3)
- As I need 100MB, I allow air above for the file (110M) and a bit more for the whole post (120M).

up
down
0
***Phil Ciebiera* ¶**
4 years ago

On a Microsoft platform utilizing IIS, you may run into a situation where, upon moving the uploaded file, anonymous web users can't access the content without being prompted to authenticate first...

The reason for this is, the uploaded file will inherit the permissions of the directory specified in the directive upload_tmp_dir of php.ini. If this directive isn't set, the default of C:\Windows\Temp is used.

You can work around this by granting the IUSR_[server name] user read access to your temporary upload directory, so that after you move_uploaded_file the permissions will already be set properly.

It's also a good idea to set the Execute Permissions of the upload directory to NOT include Executables, for security reasons.
To accomplish this:
-Open the IIS Manager
-Browse to the relevant sites directory where the uploads will be placed
-Right Click the folder and select Properties
-In the Directory tab of the resulting dialog, set the Execute permissions to be None

This took me a while to figure out, so I hope this helps save some other peoples time.
up
down
0
damien from valex ¶
5 years ago
This is simpler method of checking for too much POST data (alternative to that by v3 from sonic-world.ru).

```
<?php
    if ($_SERVER['REQUEST_METHOD'] == 'POST' && empty($_POST) &&
$_SERVER['CONTENT_LENGTH'] > 0) {
        throw new Exception(sprintf('The server was unable to handle that much POST data
(%s bytes) due to its current configuration', $_SERVER['CONTENT_LENGTH']));
    }
?>
```
up
down
0
Rob ¶
6 years ago
You should not have any directories within your website root that has the permissions required for file upload. If you are going to do a file upload, I recommend you use the PHP FTP Functions in conjunction with your file field, that way the files are transferred to a remote FTP location separate from your server.
up
down
0
david at cygnet dot be ¶
8 years ago
If you are experiencing problems posting files from Internet Explorer to a PHP script over an SSL connection, for instance "Page can not be displayed" or empty $_FILES and $_POST arrays (described by jason 10-Jan-2006 02:08), then check out this microsoft knowledgebase article:

```
http://support.microsoft.com/?kbid=889334

This knowledgebase article explains how since service pack 2 there may be problems posting
from IE over SSL. It is worth checking whether your problem is IE specific since this is
definitely not a PHP problem!
```

up
down
1
~caetin~ (at) ~hotpop~ (dot) ~com~ ¶
10 years ago

```
From the manual:

     If no file is selected for upload in your form, PHP will return $_FILES['userfile']
['size'] as 0, and $_FILES['userfile']['tmp_name'] as none.

As of PHP 4.2.0, the "none" is no longer a reliable determinant of no file uploaded. It's
documented if you click on the "error codes" link, but you need to look at the
$_FILES['your_file']['error']. If it's 4, then no file was selected.
```

up
down
0
geert dot php at myrosoft dot com ¶
8 years ago

```
When file names do contain single quote parts of the filename are being lost.
eg.: uploading a filename
      startName 'middlepart' endName.txt
will be uploaded (and hence stored in the _Files ['userfile'] variable as
      endName.txt
skipping everything before the second single quote.
```

up
down
0
ceo at l-i-e dot com ¶
9 years ago

```
Using /var/www/uploads in the example code is just criminal, imnsho.

One should *NOT* upload untrusted files into your web tree, on any server.

Nor should any directory within your web tree have permissions sufficient for an upload to
succeed, on a shared server. Any other user on that shared server could write a PHP script
to dump anything they want in there!

The $_FILES['userfile']['type'] is essentially USELESS.
A. Browsers aren't consistent in their mime-types, so you'll never catch all the possible
combinations of types for any given file format.
B. It can be forged, so it's crappy security anyway.
```

```
One's code should INSPECT the actual file to see if it looks kosher.

For example, images can quickly and easily be run through imagegetsize and you at least
know the first N bytes LOOK like an image.  That doesn't guarantee it's a valid image, but
it makes it much less likely to be a workable security breaching file.

For Un*x based servers, one could use exec and 'file' command to see if the Operating
System thinks the internal contents seem consistent with the data type you expect.

I've had trouble in the past with reading the '/tmp' file in a file upload.  It would be
nice if PHP let me read that file BEFORE I tried to move_uploaded_file on it, but PHP
won't, presumably under the assumption that I'd be doing something dangerous to read an
untrusted file.  Fine.   One should move the uploaded file to some staging directory.
Then you check out its contents as thoroughly as you can.  THEN, if it seems kosher, move
it into a directory outside your web tree.  Any access to that file should be through a
PHP script which reads the file.  Putting it into your web tree, even with all the checks
you can think of, is just too dangerous, imnsho.

There are more than a few User Contributed notes here with naive (bad) advice.  Be wary.
```

up
down
0
therhinoman at hotmail dot com ¶
9 years ago

```
If your upload script is meant only for uploading images, you can use the image function
getimagesize() (does not require the GD image library) to make sure you're really getting
an image and also filter image types.

<?php getimagesize($file); ?>

...will return false if the file is not an image or is not accessable, otherwise it will
return an array...

<?php
$file = 'somefile.jpg';

# assuming you've already taken some other
# preventive measures such as checking file
# extensions...

$result_array = getimagesize($file);

if ($result_array !== false) {
    $mime_type = $result_array['mime'];
    switch($mime_type) {
        case "image/jpeg":
            echo "file is jpeg type";
            break;
        case "image/gif":
            echo "file is gif type";
            break;
        default:
            echo "file is an image, but not of gif or jpeg type";
    }
} else {
    echo "file is not a valid image file";
}
?>
```

using this function along with others mentioned on this page, image ploading can be made pretty much fool-proof.

See http://php.net/manual/en/function.getimagesize.php for supported image types and more info.

up
down
0
olijon, iceland ¶
10 years ago

When uploading large images, I got a "Document contains no data" error when using Netscape and an error page when using Explorer. My server setup is RH Linux 9, Apache 2 and PHP 4.3.

I found out that the following entry in the httpd.conf file was missing:

```
<Files *.php>
  SetOutputFilter PHP
  SetInputFilter PHP
  LimitRequestBody 524288 (max size in bytes)
</Files>
```

When this had been added, everything worked smoothly.

- Oli Jon, Iceland

up
down
0
brion at pobox dot com ¶
10 years ago

Note that with magic_quotes_gpc on, the uploaded filename has backslashes added *but the tmp_name does not*. On Windows where the tmp_name path includes backslashes, you *must not* run stripslashes() on the tmp_name, so keep that in mind when de-magic_quotes-izing your input.

up
down
1

am at netactor dot NO_SPAN dot com ¶

12 years ago

```
Your binary files may be uploaded incorrectly if you use modules what recode characters.
For example, for Russian Apache, you should use
<Files ScriptThatReceivesUploads.php>
CharsetDisable On
</Files>
```

up

down

-1

maya_gomez ~ at ~ mail ~ dot ~ ru ¶

10 years ago

```
when you upload the file, $_FILES['file']['name'] contains its original name converted
into server's default charset.
if a name contain characters that aren't present in default charset, the conversion fails
and the $_FILES['file']['name'] remains in original charset.

i've got this behavior when uploading from a windows-1251 environment into koi8-r. if a
filename has the number sign "�" (0xb9), it DOES NOT GET CONVERTED as soon as there is no
such character in koi8-r.

Workaround i use:

<?php
if (strstr ($_FILES['file']['name'], chr(0xb9)) != "")
{
    $_FILES['file']['name'] = iconv (
        "windows-1251",
        "koi8-r",
        str_replace (chr(0xb9), "N.", $_FILES['file']['name']));
};
?>
```

up

down

-1

garyds at miraclemedia dot ca ¶

11 years ago

```
As it has been mentioned, Windows-based servers have trouble with the path to move the
uploaded file to when using move_uploaded_file()... this may also be the reason copy()
works and not move_uploaded_file(), but of course move_uploaded_file() is a much better
method to use. The solution in the aforementioned note said you must use "\\" in the path,
but I found "/" works as well. So to get a working path, I used something to the effect
of:

"g:/rootdir/default/www/".$_FILES['userfile']['name']

...which worked like a charm.
```

```
I am using PHP 4.3.0 on a win2k server.

Hope this helps!
```

up
down
-2
***ov at xs4all dot nl* ¶**
11 years ago

```
This took me a few days to find out: when uploading large files with a slow connection to
my WIN2K/IIS5/PHP4 server the POST form kept timing out at exactly 5 minutes. All PHP.INI
settings were large enough to accomodate huge file uploads. Searched like hell with
keywords like "file upload php timeout script" until I realised that I installed PHP as
CGI and added that as a keyword. This was the solution:

To set the timeout value:
1. In the Internet Information Services snap-in, select the computer icon and open its
property sheets.
2. Under Master Properties, select WWW Service, and then click the Edit button
3. Click the Home Directory tab.
4. Click the Configuration button.
5. Click the Process Options tab, and then type the timeout period in the CGI Script
Timeout box.
```

up
down
-2
***travis dot lewis at amd dot com* ¶**
11 years ago

```
If you we dumb like me you installed Redhat 8.0 and kept the default install of packages
for Apache 2.0 and PHP4.2.2.  I could not upload any files larger than 512kB and all the
php directorives were set to 32MB or higher.
memory_limit = 128M
post_max_size = 64M
upload_max_filesize = 32M

And my upload web page was set to 32MB as well:
<Form ID="frmAttach" Name="frmAttach" enctype="multipart/form-data"
action="attachments.php" method="POST">
<input type="hidden" name="MAX_FILE_SIZE" value="33554432" />

However, the insiduous php.conf (/etc/httpd/conf.d/php.conf) file used by default RPM
install of Redhat httpd has a LimitRequestBody set to 512kB ("524288" ).  Adjusting this
to 32MB ("33554432") got things going for the larger files.  Here is my php.conf file in
its entirety.  Hope this helps someone.  L8er.

#
# PHP is an HTML-embedded scripting language which attempts to make it
# easy for developers to write dynamically generated webpages.
#
```

```
LoadModule php4_module modules/libphp4.so

#
# Cause the PHP interpreter handle files with a .php extension.
#
<Files *.php>
    SetOutputFilter PHP
    SetInputFilter PHP
    LimitRequestBody 33554432
</Files>

#
# Add index.php to the list of files that will be served as directory
# indexes.
#
```

up
down
-2
rnagavel at yahoo dot com dot au ¶
5 years ago

```
If $_FILES is always empty, check the method of your form.
It should be POST. Default method of a form is GET.

<form action="myaction.php">
<input type="file" name"userfile">
</form>
File will not be uploaded as default method of the form is GET.

<form action="myaction.php" method="POST">
<input type="file" name"userfile">
</form>
Files will be uploaded and $_FILES will be populated.
```

up
down
-1
jahajee ¶
6 years ago

```
hi , i was having difficulty with the upload_max_filesize , if u set the max file size
lesser than the php setting then ur script to report error will only work till this
difference between ur max set file size and the php set max size .Hence if the uploaded
file exceeds the php max file size then php end abruptly without a trace of error that is
it behaves like no file is uploaded and hence no error reported .Sure if uploading a file
is optional for a form then a user who uploads larger file will get no error and still the
form will be processed only without the file.
The method of using GET can't be used for optional uploads. Can't find help even in the
bugs .Be careful with optional uploads.
jahajee
```

up
down
-1
djot at hotmail dot com ¶
8 years ago

```
-
Be carefull with setting max_file_size via
<?php ini_get('upload_max_filesize'); ?>

ini_get might return values like "2M" which will result in non working uploads.

This was the "no no" in my case:

<?php
$form = '<input type="hidden" name="MAX_FILE_SIZE"
value=".ini_get('upload_max_filesize')." />';
?>

Files were uploaded to the server, but than there was not any upload information, not even
an error message. $_FILES was completly empty.

djot
-
```

up
down
-1
romke at romke dot nl ¶
5 years ago

```
IIS7
has a upload limit of 30000000 (about 30mb)

You can change this with the command (for 250mb):

c:\windows\system32\inetsrv\appcmd set config -section:requestFiltering
-requestLimits.maxAllowedContentLength:262144000

Or manual define it in:
%windir%\system32\inetsrv\config\applicationhost.config

Add this rule before the </requestFiltering> tag:
<requestLimits maxAllowedContentLength ="262144000" />
```

up
down
-1
myko AT blue needle DOT com ¶
8 years ago

```
Just a quick note that there's an issue with Apache, the MAX_FILE_SIZE hidden form field,
and zlib.output_compression = On.  Seems that the browser continues to post up the entire
```

```
file, even though PHP throws the MAX_FILE_SIZE error properly.  Turning zlib compression
to OFF seems to solve the issue.  Don't have time to dig in and see who's at fault, but
wanted to save others the hassle of banging their head on this one.
```

up
down
-3
dmsuperman at comcast dot net ¶
9 years ago

```
I needed a file uploader for a client a little while ago, then the client didn't want it,
so I'll share with all of you. I know I hated coding it, it was confusing (for me anyway),
but I made it fairly simple to use:

<?php
function uploader($num_of_uploads=1, $file_types_array=array("txt"),
$max_file_size=1048576, $upload_dir=""){
  if(!is_numeric($max_file_size)){
    $max_file_size = 1048576;
  }
  if(!isset($_POST["submitted"])){
    $form = "<form action='".$PHP_SELF."' method='post' enctype='multipart/form-
data'>Upload files:<br /><input type='hidden' name='submitted' value='TRUE'
id='".time()."'><input type='hidden' name='MAX_FILE_SIZE' value='".$max_file_size."'>";
    for($x=0;$x<$num_of_uploads;$x++){
      $form .= "<input type='file' name='file[]'><font color='red'>*</font><br />";
    }
    $form .= "<input type='submit' value='Upload'><br /><font color='red'>*</font>Maximum
file length (minus extension) is 15 characters. Anything over that will be cut to only 15
characters. Valid file type(s): ";
    for($x=0;$x<count($file_types_array);$x++){
      if($x<count($file_types_array)-1){
        $form .= $file_types_array[$x].", ";
      }else{
        $form .= $file_types_array[$x].".";
      }
    }
    $form .= "</form>";
    echo($form);
  }else{
    foreach($_FILES["file"]["error"] as $key => $value){
      if($_FILES["file"]["name"][$key]!=""){
        if($value==UPLOAD_ERR_OK){
          $origfilename = $_FILES["file"]["name"][$key];
          $filename = explode(".", $_FILES["file"]["name"][$key]);
          $filenameext = $filename[count($filename)-1];
          unset($filename[count($filename)-1]);
          $filename = implode(".", $filename);
          $filename = substr($filename, 0, 15).".".$filenameext;
          $file_ext_allow = FALSE;
          for($x=0;$x<count($file_types_array);$x++){
            if($filenameext==$file_types_array[$x]){
              $file_ext_allow = TRUE;
            }
          }
          if($file_ext_allow){
            if($_FILES["file"]["size"][$key]<$max_file_size){
              if(move_uploaded_file($_FILES["file"]["tmp_name"][$key],
$upload_dir.$filename)){
                echo("File uploaded successfully. - <a href='".$upload_dir.$filename."'
target='_blank'>".$filename."</a><br />");
              }else{
                echo($origfilename." was not successfully uploaded<br />");
              }
            }else{
              echo($origfilename." was too big, not uploaded<br />");
            }
          }else{
            echo($origfilename." had an invalid file extension, not uploaded<br />");
          }
        }else{
          echo($origfilename." was not successfully uploaded<br />");
        }
      }
    }
  }
}
?>

uploader([int num_uploads [, arr file_types [, int file_size [, str upload_dir ]]]]);

num_uploads = Number of uploads to handle at once.

file_types = An array of all the file types you wish to use. The default is txt only.

file_size = The maximum file size of EACH file. A non-number will results in using the
default 1mb filesize.

upload_dir = The directory to upload to, make sure this ends with a /

This functions echo()'s the whole uploader, and submits to itself, you need not do a thing
but put uploader(); to have a simple 1 file upload with all defaults.
```

up
down
-1
jedi_aka at yahoo dot com ¶
7 years ago

```
For those of you trying to make the upload work with IIS on windows XP/2000/XP Media and
```

```
alike here is a quick todo.

1) Once you have created subdirectories "uploads/"  in the same directory wher you code is
running use the code from oportocala above and to make absolutely sure sure that the file
you are trying to right is written under that folder. ( I recomend printing it using echo
$uploadfile; )

2) In windows explorer browse to the upload directory created above and share it. To do
that execute the following substeps.
     a) Right click the folder click "sharing and security..."
     b) Check 'Share this folder on the network'
     c) Check 'Allow network users to change my files' ( THIS STEP IS VERY IMPORTANT )
     d) click 'ok' or 'apply'

3) you can then go in the IIS to set read and write permissions for it. To do that execute
the followin substeps.
     a) Open IIS (Start/Controp Panel (classic View)/ Admistrative tools/Internet
Information Service
     b) Browse to your folder (the one we created above)
     c) right click and select properties.
     d) in the Directory tab, make sure, READ, WRITE, AND DIRECTORY BROWSING are checked.
     e) For the security freaks out there, You should also make sure that 'execute
permissions:' are set to Script only or lower (DO NOT SET IT TO 'script and executable)'(
that is because someone could upload a script to your directory and run it. And, boy, you
do not want that to happen).

there U go.

Send me feed back it if worked for you or not so that I can update the todo.

jedi_aka@yahoo.com

PS: BIG thanks to oportocala
```

up
down
-1
steve dot criddle at crd-sector dot com ¶
10 years ago

```
IE on the Mac is a bit troublesome.  If you are uploading a file with an unknown file
suffix, IE uploads the file with a mime type of "application/x-macbinary".  The resulting
file includes the resource fork wrapped around the file.  Not terribly useful.

The following code assumes that the mime type is in $type, and that you have loaded the
file's contents into $content.  If the file is in MacBinary format, it delves into the
resource fork header, gets the length of the data fork (bytes 83-86) and uses that to get
rid of the resource fork.

(There is probably a better way to do it, but this solved my problem):
```

```
<?php
if ($type == 'application/x-macbinary') {
    if (strlen($content) < 128) die('File too small');
    $length = 0;
    for ($i=83; $i<=86; $i++) {
        $length = ($length * 256) + ord(substr($content,$i,1));
          }
    $content = substr($content,128,$length);
}
?>
```

up
down
-2
mariodivece at bytedive dot com ¶
8 years ago

```
Just wanted to point out a detail that might be of interest to some:

when using base64_encode to store binary data in a database, you are increasing the size
of the data by 1.33 times. There is a nicer way of storing the data directly. Try the
following:

<?php $data = mysql_real_escape_string($data); ?>

This will leave the data untouched and formatted in the correct way and ready to be
inserted right into a MySQL statement without wasting space.

By the way, I'd like to thank therebechips for his excellent advice on data chunks.
```

up
down
-3
javasri at yahoo dot com ¶
9 years ago

```
On windows XP, SP2, Explorer at times fails to upload files without extensions.

$_FILES array is null in that case. Microsoft says its a security feature(!)

The only solution we could comeup is to enforce uploaded file  to have an extention.
```

up
down
-10
Matze ¶
4 months ago

```
Ahri
Cait
Mundo
Elise
draven
```

```
ezreal
eve
fizz
Jax
Jinx
Kassadin
Kayle
Khazinx
Leblanc
Leesin
Leona
Lucian
lulu
lux
nami
nasus
nid
noc
pant
renek
rengar
riven
shyvana
trundel
varus
vi
vayne
yasuo
zed
zac
zigs
```

add a note

- My PHP.net
- Contact
- Other PHP.net sites
- Mirror sites
- Privacy policy