

Generated Documentation



Contents

mysqlfs.php	0
Class mysqlfs	0
Var \$connection	0
Var \$group	0
Var \$groups	0
Var \$security_settings	0
Var \$table	0
Var \$user	0
Constructor construct	0
Method basename	0
Method chgrp	0
Method chmod	0
Method chown	0
Method copy	0
Method dirname	0
Method file	0
Method filegroup	0
Method filemtime	0
Method fileowner	0
Method fileperms	0
Method filesize	0
Method filetype	0
Method file_exists	0
Method file_get_contents	0
Method file_put_contents	0
Method getinfo	0
Method is_dir	0
Method is_empty	0
Method is_executable	0
Method is_file	0
Method is_readable	0
Method is_writable	0
Method mime_content_type	0
Method mkdir	0
Method mkfile	0
Method move_uploaded_file	0
Method readdir_full	0
Method readfile	0
Method rename	0
Method rmdir	0
Method scandir	0
Method security	0
Method setuser	0
Method simplify_path	0

Method sizetohuman	0
Method unlink	0
Method valid_group_function	0
Method valid_user_function	0
Class mysqlfs_validate	0
Method valid_group_function	0
Method valid_user_function	0
Appendix A - Class Trees	0
mysqlfs	0

mysqlfs.php

The main library file for MySQLfs

- **Package** mysqlfs

Class mysqlfs

[line 11]

This is the interface to MySQLfs

- **Package** mysqlfs
- **Author** Lars Vierbergen

mysqlfs::\$connection

mysqli = [line 37]

Holds the MySQL connection

- **Access** protected

mysqlfs::\$group

string = 0 [line 16]

The main group of the user

- **Access** public

mysqlfs::\$groups

array = array() [line 26]

All groups the user is a member of

- **Access** public

mysqlfs::\$security_settings

array = array() [line 32]

Holds the security settings

- **Deprecated** 2.0
- **Access** protected

mysqlfs::\$table

string = [line 42]

Holds the table for MySQLfs

- **Access** protected

mysqlfs::\$user

string = NULL *[line 21]*

The userid if the users

- **Access public**

Constructor *void* function `mysqlfs::__construct($conn, [$tbl = 'index'])` *[line 48]*

Function Parameters:

- *MySQLi* **\$conn** The MySQLi link to the database
- *string* **\$tbl** The name of the index table, the datatable will have 'data' appended

Startup function for the class. Registers MySQL connection and table settings

string function `mysqlfs::basename($path, [$suffix = NULL])` *[line 705]*

Function Parameters:

- *string* **\$path** A path
- *string* **\$suffix** If the name components ends in \$suffix, this will also be cut off.

Given a string containing the path to a file or directory, this function will return the trailing name component.

bool function `mysqlfs::chgrp($path, $group)` *[line 504]*

Function Parameters:

- *string* **\$path**
- *string* **\$group** New group of the file

Changes group

- Uses [mysqlfs_validate::valid_group_function\(\)](#)

bool function mysqlfs::chmod(\$path, \$mode) [*line 473*]

Function Parameters:

- *string* **\$path**
- *int* **\$mode** The new permissions for the file

Change permissions

bool function mysqlfs::chown(\$path, \$user) [*line 488*]

Function Parameters:

- *string* **\$path**
- *string* **\$user** New user of the file

Change owner

- Uses [mysqlfs_validate::valid_user_function\(\)](#)

bool function mysqlfs::copy(\$source, \$dest) [*line 552*]

Function Parameters:

- *string* **\$source** The current location of the file
- *string* **\$dest** The location you want the file to be copied to

Copy a file to another location

string function mysqlfs::dirname(\$path) [*line 695*]

Function Parameters:

- *string* **\$path** A path

Given a string containing the path of a file or directory, this function will return the parent directory's path.

array function mysqlfs::file(\$path, [\$flags = 0]) [*line 237*]

Function Parameters:

- *string* **\$path**
- *int* **\$flags**

Reads an entire file into an array.

string function mysqlfs::filegroup(\$path) [*line 123*]

Function Parameters:

- *string* **\$path**

Gets the file group.

int function mysqlfs::filemtime(\$path) [*line 133*]

Function Parameters:

- *string* **\$path**

This function returns the time when the data blocks of a file were being written to, that is, the time when the content of the file was changed.

string function mysqlfs::fileowner(\$path) [*line 142*]

Function Parameters:

- *string* **\$path**

Gets the file owner.

string function mysqlfs::fileperms(\$path) [*line 151*]

Function Parameters:

- *string* \$path

Gets permissions for the given file.

int function mysqlfs::filesize(\$path) [*line 160*]

Function Parameters:

- *string* \$path

Gets the size for the given file in bytes

string function mysqlfs::filetype(\$path) [*line 169*]

Function Parameters:

- *string* \$path

Returns the type of the given file.

bool function mysqlfs::file_exists(\$path) [*line 306*]

Function Parameters:

- *string* \$path

Checks wether a file exists

mixed function mysqlfs::file_get_contents(\$path) [*line 220*]

Function Parameters:

- *string* **\$path**

Returns contents of the file at \$path or false if \$path isn't a file

bool function mysqlfs::file_put_contents(\$path, \$data) [*line 372*]

Function Parameters:

- *string* **\$path** The path to write to
- *string* **\$data** The data to write to the file

Writes data to the file

mixed function mysqlfs::getinfo(\$path, [\$what = NULL]) [*line 104*]

Function Parameters:

- *string* **\$path** The path where to retrieve the information from
- *string* **\$what** Which field to return

Gets information about a path

bool function mysqlfs::is_dir(\$path) [*line 210*]

Function Parameters:

- *string* **\$path**

Checks wether a path if a directory or not.

bool function mysqlfs::is_empty(\$path) [*line 261*]

Function Parameters:

- *string* **\$path**

Checks if a directory is empty

bool function mysqlfs::is_executable(\$path) [*line 431*]

Function Parameters:

- *string* \$path

Checks if the file is executable by php

- **Deprecated 2.0**

bool function mysqlfs::is_file(\$path) [*line 200*]

Function Parameters:

- *string* \$path

Checks whether a path is a file or not.

bool function mysqlfs::is_readable(\$path) [*line 387*]

Function Parameters:

- *string* \$path

Checks whether a file is readable

bool function mysqlfs::is_writable(\$path) [*line 402*]

Function Parameters:

- *string* \$path

Checks if the file is writeable

string function mysqlfs::mime_content_type(\$path) [*line 357*]

Function Parameters:

- *string* **\$path**

Gets the content type based on the extention

bool function mysqlfs::mkdir(\$path, [\$mode = NULL]) [*line 322*]

Function Parameters:

- *string* **\$path**
- *int* **\$mode** Octal permissions of the file. If not defined, it takes the permissions of the parent directory

Creates a directory

bool function mysqlfs::mkfile(\$path, [\$mode = NULL], [\$mime = NULL]) [*line 340*]

Function Parameters:

- *string* **\$path**
- *int* **\$mode** Octal permissions of the file. If not defined, it takes the permissions of the parent directory
- *string* **\$mime** The mimetype of the file, if not defined, the mimetype will be looked up by extention

Creates a file

bool function mysqlfs::move_uploaded_file(\$source, \$dest) [*line 589*]

Function Parameters:

- *string* **\$source** The source path on the real filesystem
- *string* **\$dest** The destination on MySQLfs

Moves a newly uploaded file to its final location

array function mysqlfs::readdir_full(\$path) [*line 72*]

Function Parameters:

- *string* **\$path** The path to read

Reads the directory into an array

int function mysqlfs::readfile(\$path) [*line 600*]

Function Parameters:

- *string* **\$path**

Reads a file and writes it to the output buffer

bool function mysqlfs::rename(\$oldpath, \$newpath) [*line 520*]

Function Parameters:

- *string* **\$oldpath** The current location of the file
- *string* **\$newpath** The new location of the file

Rename a file

bool function mysqlfs::rmdir(\$path) [*line 276*]

Function Parameters:

- *string* **\$path**

Removes a directory

array function mysqlfs::scandir(\$path, [\$sorting_order = 0]) [*line 657*]

Function Parameters:

- *string* **\$path**

- *int* **\$sorting_order** sort ascending if omitted or 0, descending if 1

Returns an array of files and directories from the directory.

mixed function mysqlfs::security(\$key) [*line 450*]

Function Parameters:

- *string|array* **\$key**

You can add some extra security here

To write (once) supply array with key-value pairs To read, supply the array key

- **Deprecated 2.0**

void function mysqlfs::setuser([\$user = NULL], [\$group = 0], [\$groups = array()]) [*line 62*]

Function Parameters:

- *string* **\$user** The identifier of the current user
- *string* **\$group** The identifier of the main group of the user
- *array* **\$groups** All groups the user is member of

Sets \$user and \$group (user's main group) and \$groups (all groups user is member of)

string function mysqlfs::simplify_path(\$path) [*line 179*]

Function Parameters:

- *string* **\$path**

Removes ./ and resolves ../

string function mysqlfs::sizetohuman(\$size) [*line 92*]

Function Parameters:

- *int* **\$size** The size in bytes to convert

Converts bytes to human-readable file sizes

bool function mysqlfs::unlink(\$path) [*line 291*]

Function Parameters:

- *string* **\$path**

Removes a file

bool function mysqlfs::valid_group_function(\$group) [*line 726*]

Function Parameters:

- *mixed* **\$group**

Prevents errors when no validating class is provided

bool function mysqlfs::valid_user_function(\$user) [*line 718*]

Function Parameters:

- *mixed* **\$user**

Prevents errors when no validating class is provided

Class mysqlfs_validate

[*line 736*]

This interface is used to construct user and group checking classes

- **Package** mysqlfs
- **Author** Lars Vierbergen

bool function mysqlfs_validate::valid_group_function(\$group) [*line 748*]

Function Parameters:

- *string* **\$group** The group to check

This function checks wether a group is valid

- **Usedby** [mysqlfs::chgrp\(\)](#)

bool function mysqlfs_validate::valid_user_function(\$user) [*line 742*]

Function Parameters:

- *string* **\$user** The user to check

This function checks wether a user is valid

- **Usedby** [mysqlfs::chown\(\)](#)

Appendix A - Class Trees

Package mysqlfs

mysqlfs

- [mysqlfs](#)

mysqlfs_validate

- [mysqlfs_validate](#)

Index

C

constructor mysqlfs::__construct()	0
<i>Startup function for the class. Registers MySQL connection and table settings</i>	

M

mysqlfs::mkfile()	0
<i>Creates a file</i>	
mysqlfs::mkdir()	0
<i>Creates a directory</i>	
mysqlfs::move_uploaded_file()	0
<i>Moves a newly uploaded file to its final location</i>	
mysqlfs::readdir_full()	0
<i>Reads the directory into an array</i>	
mysqlfs::readfile()	0
<i>Reads a file and writes it to the output buffer</i>	
mysqlfs::mime_content_type()	0
<i>Gets the content type based on the extension</i>	
mysqlfs::is_writable()	0
<i>Checks if the file is writeable</i>	
mysqlfs::is_empty()	0
<i>Checks if a directory is empty</i>	
mysqlfs::is_executable()	0
<i>Checks if the file is executable by php</i>	
mysqlfs::is_file()	0
<i>Checks whether a path is a file or not.</i>	
mysqlfs::is_readable()	0
<i>Checks whether a file is readable</i>	
mysqlfs::rename()	0
<i>Rename a file</i>	
mysqlfs::rmdir()	0
<i>Removes a directory</i>	
mysqlfs::valid_user_function()	0
<i>Prevents errors when no validating class is provided</i>	
mysqlfs::valid_group_function()	0
<i>Prevents errors when no validating class is provided</i>	
mysqlfs_validate	0
<i>This interface is used to construct user and group checking classes</i>	
mysqlfs_validate::valid_group_function()	0
<i>This function checks whether a group is valid</i>	
mysqlfs_validate::valid_user_function()	0
<i>This function checks whether a user is valid</i>	
mysqlfs::unlink()	0
<i>Removes a file</i>	

mysqlfs::sizetohuman()	0
<i>Converts bytes to human-readable file sizes</i>	
mysqlfs::scandir()	0
<i>Returns an array of files and directories from the directory.</i>	
mysqlfs::security()	0
<i>You can add some extra security here</i>	
mysqlfs::setuser()	0
<i>Sets \$user and \$group (user's main group) and \$groups (all groups user is member of)</i>	
mysqlfs::simplify_path()	0
<i>Removes ../ and resolves ../</i>	
mysqlfs::is_dir()	0
<i>Checks whether a path is a directory or not.</i>	
mysqlfs::getinfo()	0
<i>Gets information about a path</i>	
mysqlfs::basename()	0
<i>Given a string containing the path to a file or directory, this function will return the trailing name component.</i>	
mysqlfs::\$user	0
<i>The userid of the users</i>	
mysqlfs::chgrp()	0
<i>Changes group</i>	
mysqlfs::chmod()	0
<i>Change permissions</i>	
mysqlfs::chown()	0
<i>Change owner</i>	
mysqlfs::\$table	0
<i>Holds the table for MySQLfs</i>	
mysqlfs::\$security_settings	0
<i>Holds the security settings</i>	
mysqlfs	0
<i>This is the interface to MySQLfs</i>	
mysqlfs::\$connection	0
<i>Holds the MySQL connection</i>	
mysqlfs::\$group	0
<i>The main group of the user</i>	
mysqlfs::\$groups	0
<i>All groups the user is a member of</i>	
mysqlfs::copy()	0
<i>Copy a file to another location</i>	
mysqlfs::dirname()	0
<i>Given a string containing the path of a file or directory, this function will return the parent directory's path.</i>	
mysqlfs::filetype()	0
<i>Returns the type of the given file.</i>	
mysqlfs::file_exists()	0
<i>Checks whether a file exists</i>	
mysqlfs::file_get_contents()	0
<i>Returns contents of the file at \$path or false if \$path isn't a file</i>	
mysqlfs::file_put_contents()	0
<i>Writes data to the file</i>	
mysqlfs::filesize()	0
<i>Gets the size for the given file in bytes</i>	
mysqlfs::fileperms()	0

mysqlfs::file()	Gets permissions for the given file.	0
mysqlfs::filegroup()	Reads an entire file into an array.	0
mysqlfs::filetime()	Gets the file group.	0
mysqlfs::fileowner()	This function returns the time when the data blocks of a file were being written to, that is, the time when the content of the file was changed.	0
mysqlfs.php	Gets the file owner.	0
	The main library file for MySQLfs	0