WebDAV

Module: mod_webdav

Description

Table of contents
WebDAV
Description
Options
Examples
symlinks
client access
litmus tests
Limitations
Installation
PUT, DELETE, COPY, MOVE, MKCOL
PROPFIND, PROPPATCH
LOCK, UNLOCK

The WebDAV module is an implementation of RFC 4918. (since 1.4.54)

Supported methods:

- PROPFIND
- PROPPATCH
- OPTIONS
- MKCOL
- COPY
- MOVE
- DELETE
- PUT
- LOCK
- UNLOCK

and the usual GET, POST, HEAD from HTTP/1.1.

Mounting an open-access WebDAV resource into Windows 10, Windows XP (Network Places), Mac OS X (Finder) and Linux (Konqueror) works. Mounting an authenticated WebDAV resource works in Mac OS X and Linux. `litmus` tests pass since lighttpd 1.4.54.

Options

- **webdav.activate**
  Enables WebDAV functionality. If used within a config conditional in lighttpd.conf, this can be a path underneath the document root, e.g. "/dav/" URL mapping to "<document-root>/dav/". Note: mod_webdav expects to exclusively manage the WebDAV collection in the filesystem. Inconsistent results may occur if e.g. mod_alias is used on paths underneath the WebDAV root.
  Default: disable

- **webdav.is-readonly**
  Only allow reading methods (GET, PROPFIND, OPTIONS) on WebDAV resources.
  Default: disable (i.e. writable)

- **webdav.sqlite-db-name**
  The full path to the file you would like to use as your db file. This is required for webdav props and locks.
  Default: (empty)

- **webdav.opts** (since 1.4.54)
  List of additional options to configure mod_webdav.
  Default: (empty)
  `webdav.opts += ("propfind-depth-infinity" => "enable")` (default: "disable") (since 1.4.56)
  Permit PROPFIND with request header "Depth: infinity"
  `webdav.opts += ("deprecated-unsafe-partial-put" => "enable")` (default: "disable") (since 1.4.54)
  Versions of mod_webdav earlier than 1.4.54 were not RFC compliant and permitted partial PUT, which is unsafe. In lighttpd 1.4.54, this is disabled by default, and the unsafe behavior can be enabled with the following option, provided for backwards compatibilty.
  `webdav.opts += ("unsafe-propfind-follow-symlink" => "enable")` (default: "disable") (since 1.4.56)
  Instruct PROPFIND to follow symlinks if and only if `webdav.is-readonly = "enable"` is also set. WebDAV does not have a concept of symlinks. See symlinks
  `webdav.opts += ("partial-put-copy-modify" => "enable")` (default: "disable") (since 1.4.65)
  Safely implement partial-PUT as sequence of copy-modify-rename, which can be fast on modern filesystems supporting cloning file extents.

- **webdav.log-xml**
  debugging: log XML requests and responses to the error log
  Default: disable

Examples

To enable WebDAV for the /dav directory, you wrap your webdav options in a conditional. You have to use the regex like below as you want to match the directory /dav and everything below it, but not e.g. /davos.

```
$HTTP["url"] =~ "^/dav(?:/|$)" {
  dir-listing.activate = "enable"
  webdav.activate = "enable"
  webdav.is-readonly = "enable"
}
```

If you would like to modify files and enable WebDAV LOCK/UNLOCK support:

```
$HTTP["url"] =~ "^/dav(?:/|$)" {
  dir-listing.activate = "enable"
  webdav.activate = "enable"
  #webdav.is-readonly = "disable" # (default)
  webdav.sqlite-db-name = "/var/db/lighttpd/webdav.sqlite"
}
```

If you would like the WebDAV collection to be located outside the document root

```
$HTTP["url"] =~ "^/dav(?:/|$)" {
  alias.url = ("/dav" => "/path/to/dav")
  dir-listing.activate = "enable"
  webdav.activate = "enable"
  #webdav.is-readonly = "disable" # (default)
  webdav.sqlite-db-name = "/var/db/lighttpd/webdav.sqlite"
}
```

or

```
$HTTP["url"] =^ "/dav/" {
  alias.url = ("/dav" => "/path/to/dav")
  dir-listing.activate = "enable"
  webdav.activate = "enable"
  #webdav.is-readonly = "disable" # (default)
  webdav.sqlite-db-name = "/var/db/lighttpd/webdav.sqlite"
}
else $HTTP["url"] == "/dav" {
  url.redirect = ("" => "/dav/")
  url.redirect-code = 308
}
```

symlinks

WebDAV resource and collection concepts do not perfectly map one-to-one with the unix filesystem semantics. For example, WebDAV does not have the concept of symlinks. lighttpd mod_webdav delegates GET, HEAD, POST handling to other lighttpd modules, which generally do handle symlinks. However, WebDAV methods like PROPFIND may reject symlinks to a directory unless both `webdav.opts = ("unsafe-propfind-follow-symlink" => "enable")` and `webdav.is-readonly = "enable"` are set and the resulting behavior and restrictions are acceptable. (More technical details can be found in commit 6bf0b577)

When symlinks are used to map different paths into a single tree for the web server, an alternative with mod_webdav is to use mod_alias instead of symlinks, and to have a separate `webdav.sqlite-db-name` database for each of the directory targets; multiple WebDAV roots instead of a single top-level WebDAV root. mod_alias *should not* be used to map paths that are *underneath* the webdav tree, as this can lead to inconsistent results.

Another option on some OS, bind mounts (`mount --bind`) might be used to map one filesystem location under another tree.

Note: `server.follow-symlink = "disable"` ***is not*** supported by lighttpd mod_webdav. (server.follow-symlink) Isolate your services and/or disallow untrusted users from using *other* (non-WebDAV) mechanisms to upload symlinks into the webdav root.

client access

For access to a site https://www.example.com/ over TLS where WebDAV is at the root:

Windows

- Open File Explorer
- Enter https://www.example.com/ in the Location Bar (Quick Access) and press Enter
- (optional) follow similar steps after clicking Map Drive
- Note: Windows webdav mini-redirector uses the magical path `\\www.example.com@SSL\DavWWWRoot\` where DavWWWRoot is a "special" (virtual) volume name

Mac OS

- Open the Finder application
- Click on the Go menu and then on Connect to server
- Enter https://www.example.com/ and click Connect

Some clients, such as web clients written in javascript, may require Cross-Origin Resource Sharing HTTP response headers in order to be able to connect to the lighttpd WebDAV server.
See mod_setenv for how to set HTTP response headers.

litmus tests

We use http://webdav.org/neon/litmus/ (version 0.14) to verify that we support the WebDAV spec. (#1818)
Update: https://github.com/notroj/litmus/ is more current (tag version 0.17 as of this note)

Aside: the only `litmus` props test which does not pass has to do with XML validation, which libxml2 warns about but does not report as an error.
(At this moment, two additional props tests (after version 0.17) fail unless `webdav.opts += ("propfind-depth-infinity" => "enable")`, but the tests should be fixed upstream)

Sample litmus.conf to run `litmus` tests in /dev/shm (temporary; not persistent):

```
#mkdir -p /dev/shm/dav/dav /dev/shm/db
#lighttpd -D -f litmus.conf &
#cd /dev/shm  # litmus writes child.log and debug.log to current dir
#/usr/bin/litmus -k http://127.0.0.1:8080/
#kill %1       # kill lighttpd process started above
#rm -rf /dev/shm/{dav,db,child.log,debug.log}

server.document-root = "/dev/shm/dav"
server.bind = "127.0.0.1"
server.port = 8080

server.modules += ( "mod_webdav" )
webdav.activate = "enable"
webdav.sqlite-db-name = "/dev/shm/db/webdav.db"
```

Limitations

(incomplete list)

- incomplete "shared" lock support

Installation

PUT, DELETE, COPY, MOVE, MKCOL

mod_webdav is part of the 1.4.x distribution and supports GET, HEAD, OPTIONS, PUT, DELETE, COPY, MOVE, MKCOL without any additional libraries.

parsing XML-data and storing the properties. To compile mod_webdav with property-support libxml2, libxml2-devel, sqlite, sqlite-devel packages are required and you have to build with:

```
$ configure --with-webdav-props
```

LOCK, UNLOCK

LOCK and UNLOCK are necessary to get mounting of file-systems working. This involves generate unique lock-tokens. LOCK includes parsing XML again, so have to provide the dependencies of the previous section:

```
$ ./configure --with-webdav-props --with-webdav-locks
```

Files (0)

Powered by Redmine © 2006-2025 Jean-Philippe Lang