

# Watching LWP's Progress

by Flavio Poletti  
polettix@gmail.com



Whatever the module of choice, sometimes having a program that just works is not enough. If I'm automating the download of a lot of big files, wouldn't it be nice to know how the download is doing?

If I need to access anything on the web, chances are that LWP (*The World-Wide Web library for Perl*) will be able to fetch it for me. If I need a really sophisticated way to do that, with all sorts of automation for web forms and spidering, then I'll probably look for WWW::Mechanize. Even in this case, I'm actually still using one of the modules in LWP because it's built on LWP::UserAgent.

Using LWP::UserAgent normally means that all of the download happens in memory. What if I prefer to save stuff straight to the disk? And, last but not least, massive downloads or uploads sucks my bandwidth for a significant time unless my pipes are so fat that I don't care. I'd rather be in control and decide to assign less to the program and more to my browsing experience.

## ■ The basics of LWP

I can use LWP or WWW::Mechanize to do all that wonderful stuff for web automation. Downloads can be as easy as telling the user-agent the address I want to fetch:

```
use LWP::UserAgent;
my $ua = LWP::UserAgent->new();
my $r = $ua->get(
    'http://www.theperlreview.com/');
print $r->decoded_content()
    if $r->is_success();
```

It could be even easier using the `get()` or `getstore()` functions from LWP::Simple, but they are so simple that doing proper error checking, or taking control over the transfer process, would require a shift to LWP::UserAgent anyway.

When I upload something, I'm usually interacting with a form that accepts HTTP POST methods with the Content-type set to `multipart/form-data`:

```
my $uri =
    'http://example.com/uploader.pl';
my %form_parameters = (
    name => 'you',
    email => 'you@example.com',
    file => [ '/etc/password' ],
);
```

```
my $r = $ua->post(
    $uri,
    \%form_parameters,
    'Content-Type' => 'form-data',
    # multipart/ added automatically
);
```

Of course the form parameters could be different—in particular, nobody but the form source can tell you how the file upload field is named!

## ■ Saving to a file

When I call `get()` (or `post()`) in a LWP::UserAgent object, I can pass a bunch of headers that it will send to the server in the request. Some of these headers can be *faux headers* that start with a semicolon. One of these faux headers is `:content_file`:

```
my $r = $ua->get(
    'http://example.com/loooong.tar.gz',
    ':content_file' => '/path/to/file.tar.gz'
);
```

This way, the user-agent doesn't keep the downloaded contents in the HTTP::Response object `$r`, sucking up my precious memory, but saves it in the specified path as long as bytes arrive.

## ■ Adding feedback to downloads

If my program gives the users the joys of the command line, I can add some hints that it's actually working, instead of being seemingly stuck in some infinite loop, with a progress bar:

```
$ua->show_progress('true value');
```

If I do this, I'll get a nice percentage feedback that shows what I am downloading and how far along it is:

```
** GET http://example.com/loooong.tar.gz
==> 41%
```

If I need more bells and whistles I can dive into the transfer pipeline and insert some code. I'm going to use `Term::ProgressBar` as my tool of choice to give nicely formatted feedback in the shape of a progress bar on the terminal.