

IRC HACKS™

100 Industrial-Strength Tips & Tools



O'REILLY®

Paul Mutton

HACK
#66

Feed Syndicated RSS News into IRC Channels

Have an IRC bot bring home the latest stories from an RSS feed instead of visiting your favorite news web sites every hour.

I like RSS, you like RSS, everyone and their weblogs' visitors like RSS, as well as their favorite news site. Even if you don't know what it is, you'll probably like it when you find out. The abbreviation stands for *RDF Site Summary* (or *Really Simple Syndication*, depending on who you believe), and it is basically a special kind of XML document that is commonly used to describe the latest items appearing on a web site.

This means that you could find the latest software releases from a Freshmeat RSS feed, while Slashdot's RSS feed would keep you informed about the latest published stories. But how does one use it? RSS is utilized behind the scenes of most of the "headlines" sidebars found on a growing number of web pages. People use various dedicated RSS readers to keep track of news and blog entries. *irssi* users can have the latest headlines of chosen RSS feeds cycling in their status bar.

Well, the natural next step would be to announce the latest RSS headlines in an IRC channel, so that people can hang around and watch the show, enjoying the massive information stream. So let's take that step right now with our hack, exploring RSS basics in Perl as well as practicing our usage with `Net::IRC`.

The Code

The hack will use the `Net::IRC` module for the IRC connectivity. It suffers from a horrible incompleteness of documentation, but this is perhaps beneficial for beginner Perl programmers, as they will be forced to dive into the module's source code and absorb it. That will eventually improve your coding skills, and you will get a new surge of experience with How Others Do Things.

For RSS feed fetching and munching, you can use `LWP::UserAgent` in connection with the `XML::RSS` module. In order to refresh the RSS feed every `$refresh` seconds, you must set up an alarm that will raise the `SIGALRM` signal after the given time interval. Otherwise, the code should be pretty simple and straightforward.

Save the following as `rssbot.pl`:

```
#!/usr/bin/perl -w
# Petr Baudis (c) 2004, public domain
# Slightly inspired by Stefan "tommie" Tomanek's newslines.pl.
# RSS->IRC gateway
```

```
use strict;

### Configuration section.
# In our example setup, we are going to deliver Slashdot headlines.
use vars qw ($nick $server $port $channel $rss_url $refresh);

$nick = 'slashrss';
$server = 'irc.freenode.net';
$port = 6667;
$channel = '#irchacks';
$rss_url = 'http://www.slashdot.org/slashdot.rss';
$refresh = 30*60; # in seconds; Slashdot allows refresh max. once per 30
minutes

### Preamble.
use POSIX;
use Net::IRC;
use LWP::UserAgent;
use XML::RSS;

### Connection initialization.
use vars qw ($irc $conn);
$irc = new Net::IRC;
print "Connecting to server ".$server.":".$port." with nick ".$nick."...\n";
$conn = $irc->newconn (Nick => $nick, Server => $server, Port => $port,
                    Ircname => 'RSS->IRC gateway IRC hack');

# Connect event handler - we immediately try to join our channel.
sub on_connect {
    my ($self, $event) = @_;
    print "Joining channel ".$channel."...\n";
    $self->join ($channel);
}
$conn->add_handler ('welcome', \&on_connect);

# Joined the channel, so log that.
sub on_joined {
    my ($self, $event) = @_;
    print "Joined channel ".$channel."...\n";
}
$conn->add_handler ('endofnames', \&on_joined);

# It is a good custom to reply to the CTCP VERSION request.
sub on_cversion {
    my ($self, $event) = @_;
    $self->ctcp_reply ($event->nick, 'VERSION RSS->IRS gateway IRC hack');
}
$conn->add_handler ('cversion', \&on_cversion);

### The RSS feed
use vars qw (@items);
# Fetches the RSS from server and returns a list of RSS items.
sub fetch_rss {
```

```

my $ua = LWP::UserAgent->new (env_proxy => 1, keep_alive => 1, timeout =>
30);
my $request = HTTP::Request->new('GET', $rss_url);
my $response = $ua->request ($request);
return unless ($response->is_success);
my $data = $response->content;
my $rss = new XML::RSS ();
$rss->parse($data);
foreach my $item (@{$rss->{items}}) {
    # Make sure to strip any possible newlines and similiar stuff.
    $item->{title} =~ s/\s/ /g;
}
return @{$rss->{items}};
}

# Attempts to find some newly appeared RSS items.
sub delta_rss {
    my ($old, $new) = @_;

    # If @$old is empty, it means this is the first run and
    we will therefore not do anything.
    return () unless ($old and @$old);

    # We take the first item of @$old and find it in @$new.
    Then anything before its position in @$new are the
    newly appeared items which we return.
    my $sync = $old->[0];

    # If it is at the start of @$new, nothing has changed.
    return () if ($sync->{title} eq $new->[0]->{title});

    my $item;
    for ($item = 1; $item < @$new; $item++) {
        # We are comparing the titles which might not be 100% reliable but RSS
        # streams really should not contain multiple items with same title.
        last if ($sync->{title} eq $new->[$item]->{title});
    }
    return @$new[0 .. $item - 1];
}

# Check RSS feed periodically.
sub check_rss {
    my (@new_items);
    print "Checking RSS feed [". $rss_url ."]...\n";
    @new_items = fetch_rss ();
    if (@new_items) {
        my @delta = delta_rss (\@items, \@new_items);
        foreach my $item (reverse @delta) {
            $conn->privmsg ($channel, "''. $item->{title}.'" :: '. $item->{link});
        }
        @items = @new_items;
    }
    alarm $refresh;
}

```

```
$SIG{ALRM} = \&check_rss;  
check_rss();  
  
# Fire up the IRC loop.  
$irc->start;
```

Running the Hack

Before running the script, make sure that the configuration suits your needs. Be especially careful with the `$refresh` value—check the target site first to see if it has a recommended or allowed refresh rate. For example, Slashdot currently does not allow a shorter refresh rate than 30 minutes and is happy to block your access for 72 hours if you violate this rule.

The script takes no additional arguments; you just execute it and watch the results on IRC. The script logs all its interesting activities to `stdout`, so that you have an idea what it does right now.

You can invoke this Perl script on the command line:

```
% perl rssbot.pl
```

Hacking the Hack

A similar script is available for *irssi*, which hooks only to the user interface and provides no IRC announcement functionality. It shouldn't be too difficult to add this feature if you adapt the way it announces new stories in your current window. It even supports multiple feeds.

On the other hand, the script presented here is pretty simple and should be easy to customize. You could obviously make the output nicer, perhaps by making it a bit more colorful and easier to take in, as well as providing some more details. You could make it emit the item description or add a timestamp to the message or so on. In particular, one obvious enhancement for the Slashdot newsfeed would be to display the editor's name and department information.

One caveat here is the character set you choose to use. There is no single rule specifying which charset is to be used on IRC, so the safest thing to do is obviously to go with pure ASCII. This is, of course, not always acceptable, so you usually get to choose between a national charset (frequently ISO-8859-n) and UTF-8. Opinions and customs differ wildly between IRC networks, channels, and single individuals, so it is certainly wise to leave that as an option. The XML input will usually be in UTF-8 format, so it is up to you whether you leave it alone or encode it as a different charset. Some trivial charset remapping can be done by `Unicode::String`, while for a reasonable range of target charsets, you will probably have to use `Text::Iconv`.

It is always nice to be able to control the script from IRC. Consider implementing a generic module as a `Net::IRC` extension that would provide a kind of universal administration interface for an IRC bot, and then embed the module into this script. Then you could let users subscribe to the news individually, connect various channels with various RSS feeds, and do other similar grand things.

You may now be wondering what other useful RSS feeds exist. Yes, there are all the popular geek sites and your friends' weblogs, but aside from that, how about trying something a bit more unusual? For example, the Commits Information Agency (<http://cia.navi.cx>) provides RSS feeds of the latest commits for a large number of open source projects.

—*Petr Baudis*