NAME

feed2exec – The programmable feed reader

SYNOPSIS

feed2exec {add,ls,rm,fetch,import,export}

DESCRIPTION

This command will take a configured set of feeds and fire specific plugin for every new item found in the feed.

OPTIONS

--version
Show the version and exit.

--loglevel
choose specific log level [default: WARNING]

-v, **--verbose**
show what is happening (loglevel: VERBOSE)

-d, **--debug**
show debugging information (loglevel: DEBUG)

--syslog *LEVEL*
send LEVEL logs to syslog

--config *TEXT*
configuration directory

-h, **--help**
Show this message and exit.

EXAMPLES

Simple run with no side effects:

```
feed2exec parse https://www.nasa.gov/rss/dyn/breaking_news.rss --output echo --ar
```

Saving feed items to a Maildir folder:

```
feed2exec add "NASA breaking news" https://www.nasa.gov/rss/dyn/breaking_news.rss
feed2exec fetch
```

This creates the equivalent of this configuration file in **~/.config/feed2exec/feed2exec.ini**:

```
[DEFAULT]
output = feed2exec.plugins.maildir
mailbox = '~/Maildir'

[NASA breaking news]
folder = nasa
url = https://www.nasa.gov/rss/dyn/breaking_news.rss
```

Send new feed items to Transmission:

```
feed2exec add "Example torrent list" http://example.com/torrents/feed --output tr
```

Send new feed items to Mastodon, using the *toot* commandline client:

```
feed2exec add "My site" http://example.com/blog/feed --output exec --args 'toot p
```

Send new feed items to Twitter, using the tweet commandline client from *python−twitter*:

```
feed2exec add "My site on twitter" http://example.com/blog/feed --output exec --a
```

Show feed contents:

```
feed2exec add "NASA breaking news" https://www.nasa.gov/rss/dyn/breaking_news.rss
feed2exec fetch
```

COMMANDS

- parse:

```
parse URL
    [--output PLUGIN [--args ARG [ARG [...]]]
    [--filter PLUGIN] [--filter_args ARG [ARG [...]]]
    [--mailbox PATH] [--folder PATH]
```

  The parse command loads and parses a single feed, without touching the database. This is similar to calling *add* then *fetch* on a single feed, but the feed is not kept in the configuration. This is designed to make quick tests with a new feed. The arguments are the same as the *add* command.

- fetch:

```
fetch [--parallel | -p | --jobs N | -j N] [--force | -f] [pattern]
```

  The fetch command iterates through all the configured feeds or those matching the **pattern** substring if provided.

  −−force
  skip reading and writing the cache and will consider all entries as new

  −−catchup
  do not run output plugins, equivalent of setting the output plugin to **feed2exec.plugins.null**

  −−parallel
  run parsing in the background to improve performance

  −−jobs *N*
  run N tasks in parallel maximum. implies **−−parallel** which defaults to the number of CPUs detected on the machine

- add:

```
add NAME URL
    [--output PLUGIN [--args ARG [ARG [...]]]
    [--filter PLUGIN] [--filter_args ARG [ARG [...]]]
    [--mailbox PATH] [--folder PATH]
```

  The add command adds the given feed **NAME** that will be fetched from the provided **URL**.

  −−output *PLUGIN*
  use PLUGIN as an output module. defaults to **maildir** to store in a mailbox. use **null** to just fetch the feed without fetching anything. Modules are searched in the *feed2exec.plugins* package unless the name contains a dot in which case the whole Python search path is used.

  −−args *ARGS*
  pass arguments ARGS to the output module. supports interpolation of feed parameters using, for example **{title}**

--filter *PLUGIN*
filter feed items through the PLUGIN filter plugin

--filter_args *A*
arguments passed to the filter plugin

--mailbox *PATH*
folder to store email into, defaults to **~/Maildir**.

--folder *PATH*
subfolder to store the email into

Those parameters are documented more extensively in their equivalent settings in the configuration file, see below.

- ls:

  The **ls** command lists all configured feeds as JSON packets.

- rm:

  ```
  rm NAME
  ```

  Remove the feed named **NAME** from the configuration.

- import:

  ```
  import PATH
  ```

  Import feeds from the file named PATH. The file is expected to have **outline** elements and only the **title** and **xmlUrl** elements are imported, as **NAME** and **URL** parameters, respectively.

- export:

  ```
  export PATH
  ```

  Export feeds into the file named PATH. The file will use the feed NAME elements as **title** and the URL as **xmlUrl**.

FILES

Configuration file

Any files used by feed2exec is stored in the config directory, in **~/.config/feed2exec/** or **$XDG_CONFIG_HOME/feed2exec**. It can also be specified with the **--config** commandline parameter. The main configuration file is called **feed2exec.ini**. This is an example configuration snippet:

```
[NASA breaking news]
url = https://www.nasa.gov/rss/dyn/breaking_news.rss
output = feed2exec.plugins.echo
args = {title} {link}
```

Naturally, those settings can be changed directly in the config file. Note that there is a **[DEFAULT]** section that can be used to apply settings to all feeds. For example, this will make all feeds store new items in a maildir subfolder:

```
[DEFAULT]
output = feed2exec.plugins.maildir
folder = feeds
```

This way individual feeds do not need to be individually configured.

NOTE:

feed2exec does not take care of adding the folder to "subscriptions" in the mailbox. it is assumed that folders are auto−susbcribed or the user ignores subscription. if that is a problem, you should subscribe to the folder by hand in your email client when you add a new config. you can also subscribe to a folder (say **feeds** above) directly using the **doveadm mailbox subscribe feeds** command in Dovecot, for example.

The following configuration parameters are supported:

name Human readable name for the feed. Equivalent to the **NAME** argument in the **add** command.

url Address to fetch the feed from. Can be HTTP or HTTPS, but also **file://** resources for test purposes.

output Output plugin to use. Equivalent to the **−−output** option in the **add** command.

args Arguments to pass to the output plugin. Equivalent to the **−−args** option in the **add** command.

filter Filter plugin to use. Equivalent to the **−−filter** option in the **add** command.

mailbox
Store emails in that mailbox prefix. Defaults to **˜/Maildir**.

folder Subfolder to use when writing to a mailbox. By default, a *slugified* version of the feed name (where spaces and special character are replaced by **−**) is used. For example, the feed named "NASA breaking news" would be stored in **˜/Maildir/nasa−breaking−news/**.

catchup
Skip to the latest feed items. The feed is still read and parsed, and new feed items are added to the database, but output plugins are never called.

pause Completely skip feed during fetch or parse. Similar to catchup, but doesn't fetch the feed at all and doesn't touch the cache.

Here is a more complete example configuration with all the settings used:

```
# this section will apply to all feeds
[DEFAULT]
# special folder location for maildir. I use this when I have multiple
# accounts synchronized with Offlineimap
mailbox = ˜/Maildir/Remote/

# a feed to store NASA breaking news entry in a "nasa" subfolder
# this also demonstrates the droptitle filter
[NASA breaking news]
url = https://www.nasa.gov/rss/dyn/breaking_news.rss
folder = nasa
filter = feed2exec.plugins.droptitle
filter_args = trump

# some maildir storage require dots to get subfolders. for example,
# this will store messages in INBOX/feeds/images/ on Dovecot
[NASA image of the day]
url = https://www.nasa.gov/rss/dyn/lg_image_of_the_day.rss
folder = .feeds.images

# same feed, but save to wayback machine
[NASA IOTD wayback]
url = https://www.nasa.gov/rss/dyn/lg_image_of_the_day.rss
output = feed2exec.plugins.wayback
```

```
# this demonstrates the emptysummary filter, which fixes GitHub feeds
# that lack a proper summary
[restic]
url = https://github.com/restic/restic/tags.atom
filter = feed2exec.plugins.emptysummary

# saving to a mbox folder, one file per feed instead of one file per item
[International Space Station Reports]
url = http://blogs.nasa.gov/stationreport/feed/
mailbox = ~/Mail/
folder = stationreport.mbx

# simple generic exec call example: check for broken links using linkchecker
[NASA linkchecker]
url = https://www.nasa.gov/rss/dyn/breaking_news.rss
output = feed2exec.plugins.exec
args = linkchecker --check-extern --no-robots --recursion-level 1 --quiet '{item.

# same, but with a Ikiwiki RSS feed, which needs fixing
[Ikiwiki linkchecker]
url = http://ikiwiki.info/recentchanges/index.rss
output = feed2exec.plugins.exec
filter = feed2exec.plugins.ikiwiki_recentchanges
args = linkchecker --check-extern --no-robots --recursion-level 1 --quiet '{item.

# retweet hurricane news
[NASA Hurricane breaking news]
url = https://www.nasa.gov/rss/dyn/hurricaneupdate.rss
output = feed2exec.plugins.exec
args = tweet "{item.title:.40s} {item.link:.100s}"

# same, but on the mastodon network
#
# we can have multiple entries with the same URL without problems, as
# long as the feed name is different. it does mean that the feed will
# be fetched and parsed multiple times, unfortunately.
#
# this could be improved to include the '{item.summary}' and extra markup,
# for example.
[NASA Hurricane breaking news - Mastodon]
url = https://www.nasa.gov/rss/dyn/hurricaneupdate.rss
output = feed2exec.plugins.exec
# unfortunately, this will noisily report the URL of the posted link,
# which you may not want. to avoid that, encourage upstream to do the
# right thing: https://github.com/ihabunek/toot/issues/46 ... or use
# another tool listed here:
# https://github.com/tootsuite/documentation/blob/master/Using-Mastodon/Apps.md
args = toot post "{item.title} {item.link}"
# output is disabled here. feed will be fetched and parsed, but no
# toot will be sent
catchup = True

# same, but on the Pump.io network
[NASA Hurricane breaking news - Pump]
```

```
url = https://www.nasa.gov/rss/dyn/hurricaneupdate.rss
output = feed2exec.plugins.exec
args = p post note "{item.title} {item.link}"

# crude podcast client
[NASA Whats up?]
url = https://www.nasa.gov/rss/dyn/whats_up.rss
output = feed2exec.plugins.exec
# XXX: this doesn't handle errors properly: if there is a feed without
# enclosures, the whole thing will crash.
args = wget -P /srv/podcasts/nasa/ "{item.enclosures[0].href}"
# feed is paused here. feed will not be fetched and parsed at all and
# no post will be sent.
pause = True

# download torrents linked from a RSS feed
[torrents]
url = http://example.com/torrents.rss
output = feed2exec.plugins.exec
args = transmission-remote -a '{item.link}' -w '/srv/incoming'

# same thing with an actual plugin
[torrents]
url = http://example.com/torrents.rss
output = feed2exec.plugins.transmission
args = seedbox.example.com
folder = /srv/incoming
```

Cache database

The feeds cache is stored in a **feed2exec.sqlite** file. It is a normal SQLite database and can be inspected using the normal sqlite tools. It is used to keep track of which feed items have been processed. To clear the cache, you can simply remove the file, which will make the program process all feeds items from scratch again. In this case, you should use the **--catchup** argument to avoid duplicate processing. You can also use the **null** output plugin to the same effect.

LIMITATIONS

Feed support is only as good as **feedparser** library which isn't as solid as I expected. In particular, I had issues with *feeds without dates* and *without guid*.

Unit test coverage is incomplete, but still pretty decent, above 90%.

The **exec** plugin itself is not well tested and may have serious security issues.

API, commandline interface, configuration file syntax and database format can be changed until the 1.0 release is published, at which point normal *Semantic Versioning* semantics apply.

The program is written mainly targeting Python 3.5 and should work in 3.6. Python 2.7 is not supported anymore.

The SQL storage layer is badly written and is known to trigger locking issues with SQLite when doing multiprocessing. The global LOCK object could be used to work around this issue but that could mean pretty bad coupling. A good inspiration may be the *beets story about this problem*. And of course, another alternative would be to considering something like SQLalchemy instead of rolling our own ORM.

Older feed items are not purged from the database when they disappear from the feed, which may lead to database bloat in the long term. Similarly, there is no way for plugins to remove old entry that expire from the feed.

SEE ALSO

feed2exec−plugins(1), **feed2imap(1)**, **rss2email(1)**

AUTHOR

Antoine Beaupré

COPYRIGHT