NAME

Parse::DebianChangelog – parse Debian changelogs and output them in other formats

SYNOPSIS

```
use Parse::DebianChangelog;

my $chglog = Parse::DebianChangelog->init( { infile => 'debian/changelog',
                                             HTML => { outfile => 'changelog.htm
$chglog->html;

# the following is semantically equivalent
my $chglog = Parse::DebianChangelog->init();
$chglog->parse( { infile => 'debian/changelog' } );
$chglog->html( { outfile => 'changelog.html' } );

my $changes = $chglog->dpkg_str( { since => '1.0-1' } );
print $changes;
```

DESCRIPTION

Parse::DebianChangelog parses Debian changelogs as described in the Debian policy (version 3.6.2.1 at the time of this writing). See section ''SEE ALSO'' for locations where to find this definition.

The parser tries to ignore most cruft like # or /* */ style comments, CVS comments, vim variables, emacs local variables and stuff from older changelogs with other formats at the end of the file. NOTE: most of these are ignored silently currently, there is no parser error issued for them. This should become configurable in the future.

Beside giving access to the details of the parsed file via the ''data'' method, Parse::DebianChangelog also supports converting these changelogs to various other formats. These are currently:

dpkg

Format as known from **dpkg-parsechangelog** (1). All requested entries (see ''METHODS'' for an explanation what this means) are returned in the usual Debian control format, merged in one stanza, ready to be used in a *.changes* file.

rfc822

Similar to the `dpkg` format, but the requested entries are returned as one stanza each, i.e. they are not merged. This is probably the format to use if you want a machine-usable representation of the changelog.

xml

Just a simple XML dump of the changelog data. Without any schema or DTD currently, just some made up XML. The actual format might still change. Comments and Improvements welcome.

html

The changelog is converted to a somewhat nice looking HTML file with some nice features as a quick-link bar with direct links to every entry. NOTE: This is not very configurable yet and was specifically designed to be used on <http://packages.debian.org/>. This is planned to be changed until version 1.0.

METHODS

init

Creates a new object instance. Takes a reference to a hash as optional argument, which is interpreted as configuration options. There are currently no supported general configuration options, but see the other methods for more specific configuration options which can also specified to `init`.

If `infile`, `instring` or `handle` are specified (see parse), `parse()` is called from `init`. If a fatal error is encountered during parsing (e.g. the file can't be opened), `init` will not return a valid object but `undef`!

reset_parse_errors

Can be used to delete all information about errors ocurred during previous parse runs. Note that `parse()` also calls this method.

get_parse_errors

Returns all error messages from the last parse run. If called in scalar context returns a human readable string representation. If called in list context returns an array of arrays. Each of these arrays contains

1. the filename of the parsed file or `String` if a string was parsed directly
2. the line number where the error occurred
3. an error description
4. the original line

NOTE: This format isn't stable yet and may change in later versions of this module.

get_error

Get the last non-parser error (e.g. the file to parse couldn't be opened).

parse

Parses either the file named in configuration item `infile`, the string saved in configuration item `instring` or the open file handle saved in the configuration item `handle`. In the latter case, the handle can be named by using the optional configuration item `handlename`. Accepts a hash ref as optional argument which can contain configuration items.

Returns `undef` in case of error (e.g. ''file not found'', **not** parse errors) and the object if successful. If `undef` was returned, you can get the reason for the failure by calling the get_error method.

data

`data` returns an array (if called in list context) or a reference to an array of Parse::DebianChangelog::Entry objects which each represent one entry of the changelog.

This is currently merely a placeholder to enable users to get to the raw data, expect changes to this API in the near future.

This method supports the common output options described in section ''COMMON OUTPUT OPTIONS''.

dpkg

(and **dpkg_str**)

`dpkg` returns a hash (in list context) or a hash reference (in scalar context) where the keys are field names and the values are field values. The following fields are given:

Source
: package name (in the first entry)

Version
: packages' version (from first entry)

Distribution
: target distribution (from first entry)

Urgency
: urgency (highest of all printed entries)

Maintainer
: person that created the (first) entry

Timestamp
: date (see below) as a Unix timestamp

Date
: date of the (first) entry

Closes
: bugs closed by the entry/entries, sorted by bug number

Changes
: content of the the entry/entries

`dpkg_str` returns a stringified version of this hash which should look exactly like the output of **dpkg-parsechangelog** (1). The fields are ordered like in the list above.

Both methods only support the common output options described in section ''COMMON OUTPUT OPTIONS''.

dpkg_str

See dpkg.

rfc822

(and **rfc822_str**)

`rfc822` returns an array of hashes (in list context) or a reference to this array (in scalar context) where each hash represents one entry in the changelog. For the format of such a hash see the description of the ''dpkg'' method (while ignoring the remarks about which values are taken from the first entry).

`rfc822_str` returns a stringified version of this hash which looks similar to the output of dpkg-parsechangelog but instead of one stanza the output contains one stanza for each entry.

Both methods only support the common output options described in section ''COMMON OUTPUT OPTIONS''.

rfc822_str

See rfc822.

xml

(and **xml_str**)

`xml` converts the changelog to some free-form (i.e. there is neither a DTD or a schema for it) XML.

The method `xml_str` is an alias for `xml`.

Both methods support the common output options described in section ''COMMON OUTPUT OPTIONS'' and additionally the following configuration options (as usual to give in a hash reference as parameter to the method call):

outfile
: directly write the output to the file specified

xml_str

See xml.

html

(and **html_str**)

`html` converts the changelog to a HTML file with some nice features such as a quick-link bar with direct links to every entry. The HTML is generated with the help of HTML::Template. If you want to change the output you should use the default template provided with this module as a base and read the documentation of HTML::Template to understand how to edit it.

The method `html_str` is an alias for `html`.

Both methods support the common output options described in section ''COMMON OUTPUT OPTIONS'' and additionally the following configuration options (as usual to give in a hash reference as parameter to the method call):

outfile
: directly write the output to the file specified

template
: template file to use, defaults to tmpl/default.tmpl, so you most likely want to override that. NOTE: The plan is to provide a configuration file for the module later to be able to use sane defaults here.

style
: path to the CSS stylesheet to use (a default might be specified in the template and will be honoured, see the default template for an example)

print_style
: path to the CSS stylesheet to use for printing (see the notes for `style` about default values)

html_str

See html.

init_filters

not yet documented

apply_filters

not yet documented

add_filter, delete_filter, replace_filter

not yet documented

COMMON OUTPUT OPTIONS

The following options are supported by all output methods, all take a version number as value:

since
: Causes changelog information from all versions strictly later than **version** to be used.

  (works exactly like the `-v` option of dpkg-parsechangelog).

until
: Causes changelog information from all versions strictly earlier than **version** to be used.

from
: Similar to `since` but also includes the information for the specified **version** itself.

to
: Similar to `until` but also includes the information for the specified **version** itself.

The following options also supported by all output methods but don't take version numbers as values:

all
: If set to a true value, all entries of the changelog are returned, this overrides all other options. While the XML and HTML formats default to all == true, this does of course not overwrite other options unless it is set explicitly with the call.

count
: Expects a signed integer as value. Returns `value` entries from the top of the changelog if set to a positive integer, and `abs(value)` entries from the tail if set to a negative integer.

offset
: Expects a signed integer as value. Changes the starting point for `count`, either counted from the top (positive integer) or from the tail (negative integer). `offset` has no effect if `count` wasn't given as well.

Some examples for the above options. Imagine an example changelog with entries for the versions 1.2, 1.3, 2.0, 2.1, 2.2, 3.0 and 3.1.

```
           Call                        Included entries
format({ since => '2.0' })          3.1, 3.0, 2.2
format({ until => '2.0' })          1.3, 1.2
format({ from => '2.0' })           3.1, 3.0, 2.2, 2.1, 2.0
format({ to => '2.0' })             2.0, 1.3, 1.2
format({ count => 2 }               3.1, 3.0
format({ count => -2 }              1.3, 1.2
format({ count => 3,
         offset => 2 }              2.2, 2.1, 2.0
format({ count => 2,
         offset => -3 }             2.0, 1.3
format({ count => -2,
         offset => 3 }              3.0, 2.2
format({ count => -2,
         offset => -3 }             2.2, 2.1
```

Any combination of one option of `since` and `from` and one of `until` and `to` returns the intersection of the two results with only one of the options specified.

SEE ALSO

Parse::DebianChangelog::Entry, Parse::DebianChangelog::ChangesFilters

Description of the Debian changelog format in the Debian policy: <http://www.debian.org/doc/debian-policy/ch-source.html#s-dpkgchangelog>.

AUTHOR

Frank Lichtenheld, <frank@lichtenheld.de>

COPYRIGHT AND LICENSE