NAME

Lintian::Data – Lintian interface to query lists of keywords

SYNOPSIS

```
    my $keyword;
    my $list = Lintian::Data->new('type');
    if ($list->known($keyword)) {
        # do something ...
    }
    my $hash = Lintian::Data->new('another-type', qr{\s++});
    if ($hash->value($keyword) > 1) {
        # do something ...
    }
    if ($list->value($keyword) > 1) {
        # do something ...
    }
    my @keywords = $list->all;
    if ($list->matches_any($keyword)) {
        # do something ...
    }
```

DESCRIPTION

Lintian::Data provides a way of loading a list of keywords or key/value pairs from a file in the Lintian root and then querying that list. The lists are stored in the *data* directory of the Lintian root and consist of one keyword or key/value pair per line. Blank lines and lines beginning with # are ignored. Leading and trailing whitespace is stripped.

If requested, the lines are split into key/value pairs with a given separator regular expression. Otherwise, keywords are taken verbatim as they are listed in the file and may include spaces.

This module allows lists such as menu sections, doc-base sections, obsolete packages, package fields, and so forth to be stored in simple, easily editable files.

NB: By default Lintian::Data is lazy and defers loading of the data file until it is actually needed.

Interface for the CODE argument

This section describes the interface between for the CODE argument for the class method new.

The sub will be called once for each key/pair with three arguments, KEY, VALUE and CURVALUE. The first two are the key/value pair parsed from the data file and CURVALUE is current value associated with the key. CURVALUE will be `undef` the first time the sub is called with that KEY argument.

The sub can then modify VALUE in some way and return the new value for that KEY. If CURVALUE is not `undef`, the sub may return `undef` to indicate that the current value should still be used. It is not permissible for the sub to return `undef` if CURVALUE is `undef`.

Where Perl semantics allow it, the sub can modify CURVALUE and the changes will be reflected in the result. As an example, if CURVALUE is a hashref, new keys can be inserted etc.

CLASS METHODS

new(TYPE [,SEPARATOR[, CODE]])

Creates a new Lintian::Data object for the given TYPE. TYPE is a partial path relative to the *data* directory and should correspond to a file in that directory. The contents of that file will be loaded into memory and returned as part of the newly created object. On error, **new()** throws an exception.

If SEPARATOR is given, it will be used as a regular expression for splitting the lines into key/value pairs.

If CODE is also given, it is assumed to be a sub that will pre-process the key/value pairs. See the "Interface for the CODE argument" above.

A given file will only be loaded once. If **new()** is called again with the same TYPE argument, the data previously loaded will be reused, avoiding multiple file reads.

set_vendor(PROFILE)
: Specifies vendor profile. It must be set before the first data file is loaded.

INSTANCE METHODS

all()
: Returns all keywords listed in the data file as a list in original order. In a scalar context, returns the number of keywords.

matches_any(KEYWORD[, MODIFIERS])
: Returns true if KEYWORD matches any regular expression listed in the data file. The optional MODIFIERS serve as modifiers on all regexes.

known(KEYWORD)
: Returns true if KEYWORD was listed in the data file represented by this Lintian::Data instance and false otherwise.

value(KEYWORD)
: Returns the value attached to KEYWORD if it was listed in the data file represented by this Lintian::Data instance and the undefined value otherwise. If SEPARATOR was not given, the value will '1'.

DIAGNOSTICS

no data type specified
: **new()** was called without a TYPE argument.

unknown data type `%s`
: The TYPE argument to **new()** did not correspond to a file in the *data* directory of the Lintian root.

undefined value for `%s` (type: `%s`)
: The CODE argument return undef for the KEY and no previous value for that KEY was available.

FILES

LINTIAN_INCLUDE_DIR/data
: The files loaded by this module must be located in this directory. Relative paths containing a `/` are permitted, so files may be organized in subdirectories in this directory.

  Note that lintian supports multiple LINTIAN_INCLUDE_DIRs.

AUTHOR

Originally written by Russ Allbery <rra@debian.org> for Lintian.

SEE ALSO

lintian (1), <https://lintian.debian.org/manual/section−2.6.html>