NAME

Glib::OptionGroup − group of options for command line option parsing

SYNOPSIS

```
my ($verbose, $source, $filenames) = ('', undef, []);

my $entries = [
  { long_name => 'verbose',
    short_name => 'v',
    arg_type => 'none',
    arg_value => \$verbose,
    description => 'be verbose' },

  { long_name => 'source',
    short_name => 's',
    arg_type => 'string',
    arg_value => \$source,
    description => 'set the source',
    arg_description => 'source' },

  [ 'filenames', 'f', 'filename-array', \$filenames ],
];

my $context = Glib::OptionContext->new ('- urgsify your life');
$context->add_main_entries ($entries, 'C');
$context->parse ();

# $verbose, $source, and $filenames are now updated according to the
# command line options given
```

HIERARCHY

```
Glib::Boxed
+----Glib::OptionGroup
```

METHODS

optioncontext = Glib::OptionContext->new ($parameter_string)

- `$parameter_string` (string)

optiongroup = Glib::OptionGroup->new (key => value, ...)

Creates a new option group from the given key-value pairs. The valid keys are name, description, help_description, and entries. The first three specify strings while the last one, entries, specifies an array reference of option entries. Example:

```
my $group = Glib::OptionGroup->new (
              name => 'urgs',
              description => 'Urgs Urgs Urgs',
              help_description => 'Help with Urgs',
              entries => \@entries);
```

An option entry is a hash reference like this:

```
{ long_name => 'verbose',
  short_name => 'v',
  flags => [qw/reverse hidden in-main/],
  arg_type => 'none',
  arg_value => \$verbose,
  description => 'verbose desc.',
  arg_description => 'verbose arg desc.' }
```

Of those keys only long_name, arg_type, and arg_value are required. So this is a valid option entry too:

```
{ long_name => 'package-names',
  arg_type => 'string-array',
  arg_value => \$package_names }
```

For convenience, option entries can also be specified as array references containing long_name, short_name, arg_type, and arg_value:

```
[ 'filenames', 'f', 'filename-array', \$filenames ]
```

If you don't want an option to have a short name, specify undef for it:

```
[ 'filenames', undef, 'filename-array', \$filenames ]
```

`$context`**->add_group ($group)**

- `$group` (Glib::OptionGroup)

`$context`**->add_main_entries ($entries,** `$translation_domain`**)**

- `$entries` (scalar) reference to an array of option entries
- `$translation_domain` (string)

boolean = `$context`**->get_help_enabled**
`$context`**->set_help_enabled ($help_enabled)**

- `$help_enabled` (boolean)

boolean = `$context`**->get_ignore_unknown_options**
`$context`**->set_ignore_unknown_options ($ignore_unknown)**

- `$ignore_unknown` (boolean)

optiongroup = `$context`**->get_main_group**
`$context`**->set_main_group ($group)**

- `$group` (Glib::OptionGroup)

boolean = `$context`**->parse**

This method works directly on *@ARGV*.

May croak with a Glib::Error in $@ on failure.

`$group`**->set_translate_func ($func,** `$data`**=undef)**

- `$func` (scalar)
- `$data` (scalar)

`$group`**->set_translation_domain ($domain)**

- `$domain` (string)

ENUMS AND FLAGS

enum Glib::OptionArg

- 'none' / 'G_OPTION_ARG_NONE'
- 'string' / 'G_OPTION_ARG_STRING'
- 'int' / 'G_OPTION_ARG_INT'
- 'callback' / 'G_OPTION_ARG_CALLBACK'
- 'filename' / 'G_OPTION_ARG_FILENAME'
- 'string-array' / 'G_OPTION_ARG_STRING_ARRAY'
- 'filename-array' / 'G_OPTION_ARG_FILENAME_ARRAY'
- 'double' / 'G_OPTION_ARG_DOUBLE'
- 'int64' / 'G_OPTION_ARG_INT64'

flags Glib::OptionFlags

- 'hidden' / 'G_OPTION_FLAG_HIDDEN'
- 'in-main' / 'G_OPTION_FLAG_IN_MAIN'
- 'reverse' / 'G_OPTION_FLAG_REVERSE'
- 'no-arg' / 'G_OPTION_FLAG_NO_ARG'
- 'filename' / 'G_OPTION_FLAG_FILENAME'
- 'optional-arg' / 'G_OPTION_FLAG_OPTIONAL_ARG'
- 'noalias' / 'G_OPTION_FLAG_NOALIAS'

SEE ALSO

Glib, Glib::Boxed

COPYRIGHT