NAME

Glib::ParamSpec − encapsulates metadate needed to specify parameters

DESCRIPTION

Glib::ParamSpec encapsulates the metadata required to specify parameters. You will see these most often when creating new Glib::Object types; see `Glib::Type->register` and Glib::Object::Subclass.

Parameter specifications allow you to provide limits for validation as well as nicknames and blurbs to document the parameters. Blurbs show up in reference documentation such as this page or the gtk+ C API reference; i'm not really sure where the nicknames get used. The Perl bindings for the most part ignore the difference between dashes and underscores in the paramspec names, which typically find use as the actual keys for object parameters.

It's worth noting that Glib offers various sizes of integer and floating point values, while Perl really only deals with full integers and double precision floating point values. The size distinction is important for the underlying C libraries.

HIERARCHY

`Glib::ParamSpec`

METHODS

paramspec = Glib::ParamSpec->IV ($name, `$nick`**,** `$blurb`**,** `$minimum`**,** `$maximum`**,** `$default_value`**,** `$flags`**)**

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$minimum` (integer)
- `$maximum` (integer)
- `$default_value` (integer)
- `$flags` (Glib::ParamFlags)

paramspec = Glib::ParamSpec->UV ($name, `$nick`**,** `$blurb`**,** `$minimum`**,** `$maximum`**,** `$default_value`**,** `$flags`**)**

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$minimum` (unsigned)
- `$maximum` (unsigned)
- `$default_value` (unsigned)
- `$flags` (Glib::ParamFlags)

string = `$pspec`**->get_blurb**

paramspec = Glib::ParamSpec->boolean ($name, `$nick`**,** `$blurb`**,** `$default_value`**,** `$flags`**)**

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$default_value` (boolean)
- `$flags` (Glib::ParamFlags)

paramspec = Glib::ParamSpec->boxed ($name, `$nick`**,** `$blurb`**,** `$package`**,** `$flags`**)**

- `$name` (string)

- `$nick` (string)
- `$blurb` (string)
- `$package` (string) name of the class, derived from Glib::Boxed, of the objects this property will hold.
- `$flags` (Glib::ParamFlags)

paramspec = Glib::ParamSpec->char ($name, `$nick`, `$blurb`, `$minimum`, `$maximum`, `$default_value`, `$flags`)

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$minimum` (integer)
- `$maximum` (integer)
- `$default_value` (integer)
- `$flags` (Glib::ParamFlags)

scalar = `$pspec`->get_default_value

(This is the C level `g_param_value_set_default` function.)

Note that on a `Glib::Param::Unichar` the return is a single-char string. This is the same as the constructor `Glib::ParamSpec->unichar`, but it's not the same as `Glib::Object` `get_property` / `set_property`, so an `ord()` conversion is needed if passing the default value to a unichar `set_property`.

paramspec = Glib::ParamSpec->double ($name, `$nick`, `$blurb`, `$minimum`, `$maximum`, `$default_value`, `$flags`)

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$minimum` (double)
- `$maximum` (double)
- `$default_value` (double)
- `$flags` (Glib::ParamFlags)

paramspec = Glib::ParamSpec->enum ($name, `$nick`, `$blurb`, `$enum_type`, `$default_value`, `$flags`)

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$enum_type` (string)
- `$default_value` (scalar)
- `$flags` (Glib::ParamFlags)

paramspec = Glib::ParamSpec->flags ($name, `$nick`, `$blurb`, `$flags_type`, `$default_value`, `$flags`)

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)

- `$flags_type` (string)
- `$default_value` (scalar)
- `$flags` (Glib::ParamFlags)

paramflags = `$pspec`**->get_flags**

paramspec = Glib::ParamSpec->float ($name, `$nick`**,** `$blurb`**,** `$minimum`**,** `$maximum`**,** `$default_value`**,** `$flags`**)**

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$minimum` (double)
- `$maximum` (double)
- `$default_value` (double)
- `$flags` (Glib::ParamFlags)

paramspec = Glib::ParamSpec->gtype ($name, `$nick`**,** `$blurb`**,** `$is_a_type`**,** `$flags`**)**

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$is_a_type` (string or undef) The name of a class whose subtypes are allowed as values of the property. Use `undef` to allow any type.
- `$flags` (Glib::ParamFlags)

Since: glib 2.10

paramspec = Glib::ParamSpec->int ($name, `$nick`**,** `$blurb`**,** `$minimum`**,** `$maximum`**,** `$default_value`**,** `$flags`**)**

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$minimum` (integer)
- `$maximum` (integer)
- `$default_value` (integer)
- `$flags` (Glib::ParamFlags)

paramspec = Glib::ParamSpec->int64 ($name, `$nick`**,** `$blurb`**,** `$minimum`**,** `$maximum`**,** `$default_value`**,** `$flags`**)**

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$minimum` (64 bit integer)
- `$maximum` (64 bit integer)
- `$default_value` (64 bit integer)
- `$flags` (Glib::ParamFlags)

paramspec = Glib::ParamSpec->long ($name, `$nick`**,** `$blurb`**,** `$minimum`**,** `$maximum`**,** `$default_value`**,** `$flags`**)**

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$minimum` (integer)
- `$maximum` (integer)
- `$default_value` (integer)
- `$flags` (Glib::ParamFlags)

string = `$paramspec`**->get_name**
Dashes in the name are converted to underscores.

string = `$pspec`**->get_nick**
paramspec = Glib::ParamSpec->object ($name, `$nick`**,** `$blurb`**,** `$package`**,** `$flags`**)**
- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$package` (string) name of the class, derived from Glib::Object, of the objects this property will hold.
- `$flags` (Glib::ParamFlags)

paramspec = Glib::ParamSpec->override ($name, `$overridden`**)**
- `$name` (string)
- `$overridden` (Glib::ParamSpec)

Since: glib 2.4

string = `$pspec`**->get_owner_type**
paramspec = Glib::ParamSpec->param_spec ($name, `$nick`**,** `$blurb`**,** `$package`**,** `$flags`**)**
- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$package` (string) name of the class, derived from Glib::ParamSpec, of the objects this property will hold.
- `$flags` (Glib::ParamFlags)

paramspec or undef = `$pspec`**->get_redirect_target**
Since: glib 2.4

paramspec = Glib::ParamSpec->scalar ($name, `$nick`**,** `$blurb`**,** `$flags`**)**
- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$flags` (Glib::ParamFlags)

ParamSpec to be used for any generic perl scalar, including references to complex objects.

Currently `Gtk2::Builder` cannot set object properties of this type (there's no hooks for property value parsing, as of Gtk 2.20), so prefer the builtin types if buildable support for an object matters. A `boxed` of `Glib::Strv` can give an array of strings. A signal handler callback can do most of what a coderef might.

paramspec = Glib::ParamSpec->string ($name, `$nick`**,** `$blurb`**,** `$default_value`**,** `$flags`**)**
- `$name` (string)

- `$nick` (string)
- `$blurb` (string)
- `$default_value` (string or undef)
- `$flags` (Glib::ParamFlags)

paramspec = Glib::ParamSpec->uchar ($name, `$nick`, `$blurb`, `$minimum`, `$maximum`, `$default_value`, `$flags`)

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$minimum` (unsigned)
- `$maximum` (unsigned)
- `$default_value` (unsigned)
- `$flags` (Glib::ParamFlags)

paramspec = Glib::ParamSpec->uint ($name, `$nick`, `$blurb`, `$minimum`, `$maximum`, `$default_value`, `$flags`)

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$minimum` (unsigned)
- `$maximum` (unsigned)
- `$default_value` (unsigned)
- `$flags` (Glib::ParamFlags)

paramspec = Glib::ParamSpec->uint64 ($name, `$nick`, `$blurb`, `$minimum`, `$maximum`, `$default_value`, `$flags`)

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$minimum` (64 bit unsigned)
- `$maximum` (64 bit unsigned)
- `$default_value` (64 bit unsigned)
- `$flags` (Glib::ParamFlags)

paramspec = Glib::ParamSpec->ulong ($name, `$nick`, `$blurb`, `$minimum`, `$maximum`, `$default_value`, `$flags`)

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$minimum` (unsigned)
- `$maximum` (unsigned)
- `$default_value` (unsigned)
- `$flags` (Glib::ParamFlags)

paramspec = Glib::ParamSpec->unichar ($name, `$nick`**,** `$blurb`**,** `$default_value`**,** `$flags`**)**

- `$name` (string)
- `$nick` (string)
- `$blurb` (string)
- `$default_value` (character)
- `$flags` (Glib::ParamFlags)

string = `$pspec`**->get_value_type**
bool = `$paramspec`**->value_validate ($value)**
(bool, newval) = `$paramspec`**->value_validate ($value)**

- `$value` (scalar)

In scalar context return true if `$value` must be modified to be valid for `$paramspec`, or false if it's valid already. In array context return also a new value which is `$value` made valid.

`$value` must be the right type for `$paramspec` (with usual stringizing, numizing, etc). `value_validate` checks the further restrictions such as minimum and maximum for a numeric type or allowed characters in a string. The ''made valid'' return is then for instance clamped to the min/max, or offending chars replaced by a substitutor.

integer = `$pspec`**->values_cmp ($value1,** `$value2`**)**

- `$value1` (scalar)
- `$value2` (scalar)

Compares *value1* with *value2* according to *pspec*, and returns −1, 0 or +1, if value1 is found to be less than, equal to or greater than value2, respectively.

ENUMS AND FLAGS

flags Glib::ParamFlags

- 'readable' / 'G_PARAM_READABLE'
- 'writable' / 'G_PARAM_WRITABLE'
- 'construct' / 'G_PARAM_CONSTRUCT'
- 'construct−only' / 'G_PARAM_CONSTRUCT_ONLY'
- 'lax−validation' / 'G_PARAM_LAX_VALIDATION'
- 'private' / 'G_PARAM_PRIVATE'

SEE ALSO

Glib

COPYRIGHT