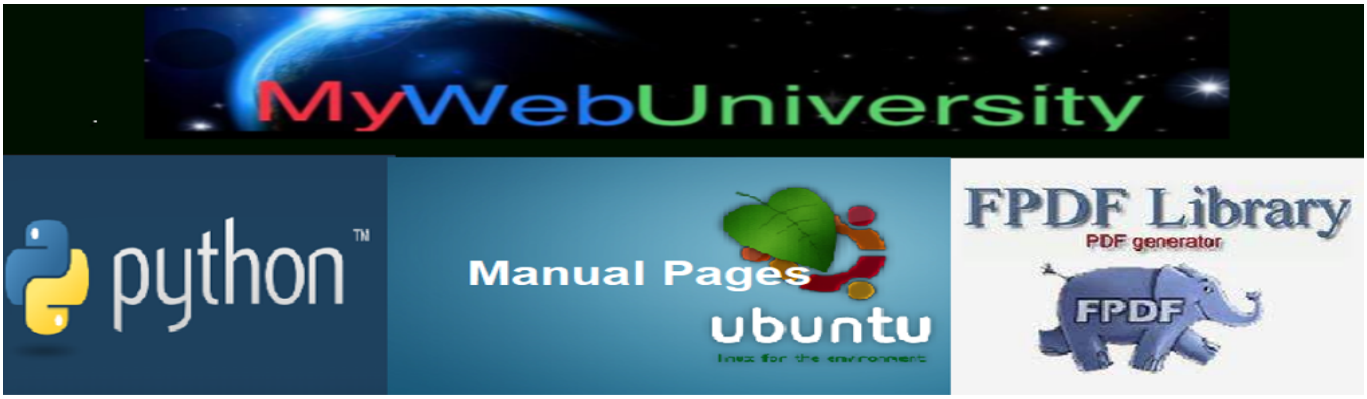

Rocky Enterprise Linux 9.2 Manual Pages on command 'Glib::Error.3pm'

$ man Glib::Error.3pm

Glib::Error(3pm) User Contributed Perl Documentation Glib::Error(3pm)

NAME

Glib::Error - Exception Objects based on GError

SYNOPSIS

```
eval {
  my $pixbuf = Gtk2::Gdk::Pixbuf->new_from_file ($filename);
  $image->set_from_pixbuf ($pixbuf);
};
if ($@) {
  print "$@\n";
  if (Glib::Error::matches ($@, 'Gtk2::Gdk::Pixbuf::Error',
                            'unknown-format')) {
    change_format_and_try_again ();
  } elsif (Glib::Error::matches ($@, 'Glib::File::Error', 'noent')) {
    change_source_dir_and_try_again ();
  } else {
    # don't know how to handle this
    die $@;
  }
}
```

DESCRIPTION

Gtk2-Perl translates GLib's GError runtime errors into Perl exceptions, by creating exception objects based on Glib::Error. Glib::Error overloads the stringification operator, so a Glib::Error object will act like a string if used with print() or warn(), so most code using $@ will not even know the difference.

The point of having exception objects, however, is that the error messages in GErrors are often localized with NLS translation. Thus, it's not good for your code to attempt to handle errors by string matching on the the error message. Glib::Error provides a way to get to the deterministic error code.

You will typically deal with objects that inherit from Glib::Error, such as Glib::Convert::Error, Glib::File::Error, Gtk2::Gdk::Pixbuf::Error, etc; these classes are provided by the libraries that define the error domains. However, it is possible to get a base Glib::Error when the bindings encounter an unknown or unbound error domain. The interface used here degrades nicely in such a situation, but in general you should submit a bug report to the binding maintainer if you get such an exception.

HIERARCHY

```
Glib::Boxed
+----Glib::Error
```

METHODS

scalar = Glib::Error::new ($class, $code, $message)

scalar = $class->new ($code, $message)

? $code (Glib::Enum) an enumeration value, depends on $class

? $message (string)

Create a new exception object of type $class, where $class is associated with a GError domain. $code should be a value from the enumeration type associated with this error domain. $message can be anything you like, but should explain what happened from the point of view of a user.

integer = $error->code

This is the numeric error code. Normally, you'll want to use "value" instead, for readability.

string = $error->domain

The error domain. You normally do not need this, as the object will be blessed into a corresponding class.

string = $error->location

The source line and file closest to the emission of the exception, in the same format that you'd get from croak() or die().

If there's non-ascii characters in the filename Perl leaves them as raw bytes, so you may have to put the string through Glib::filename_display_name for a wide-char form.

boolean = $error->matches ($domain, $code)

- $domain (string)

- $code (scalar)

Returns true if the exception in $error matches the given $domain and $code. $domain may be a class name or domain quark (that is, the real string used in C). $code may be an integer value or an enum nickname; the enum type depends on the value of $domain.

string = $error->message

The error message. This may be localized, as it is intended to be shown to a user.

Glib::Error::register ($package, $enum_package)

- $package (string) class name to register as a Glib::Error.

- $enum_package (string) class name of the enum type to use for this domain's error codes.

Register a new error domain. Glib::Error will be added @package::ISA for you.

enum_package must be a valid Glib::Enum type, either from a C library or registered with "Glib::Type::register_enum". After registering an error domain, you can create or throw exceptions of this type.

scalar = Glib::Error::throw ($class, $code, $message)

scalar = $class->throw ($code, $message)

? $code (Glib::Enum) an enumeration value, depends on $class

? $message (string)

Throw an exception with a Glib::Error exception object. Equivalent to "croak (Glib::Error::new ($class, $code, $message));".

string = $error->value

The enumeration value nickname of the integer value in "$error->code", according to this error domain. This will not be available if the error object is a base Glib::Error, because the bindings will have no idea how to get to the correct nickname.

SEE ALSO

Glib, Glib::Boxed

COPYRIGHT