



# class GTK::V3::Gtk::GtkBuilder

## Table of Contents

- 0.1 [GtkBuilder — Build an interface from an XML UI definition](#)
- 1 [Synopsis](#)
- 2 [Methods](#)
  - 2.1 [gtk\\_builder\\_new](#)
  - 2.2 [\[gtk\\_builder\\_\] new\\_from\\_file](#)
  - 2.3 [\[gtk\\_builder\\_\] new\\_from\\_string](#)
  - 2.4 [\[gtk\\_builder\\_\] add\\_from\\_file](#)
  - 2.5 [\[gtk\\_builder\\_\] add\\_from\\_string](#)
  - 2.6 [\[gtk\\_builder\\_\] get\\_object](#)
  - 2.7 [\[gtk\\_builder\\_\] get-type-from-name](#)
  - 2.8 [new](#)

```
unit class GTK::V3::Gtk::GtkBuilder;  
also is GTK::V3::Glib::GObject;
```

## GtkBuilder — Build an interface from an XML UI definition

### Synopsis

```
my GTK::V3::Gtk::GtkBuilder $builder .= new(:filename($ui-file));  
my GTK::V3::Gtk::GtkButton $start-button .= new(:build-id<startButton>);
```

Note: `GTK::Glade` is a package build around this builder class. That package is able to automatically register the signals defined in the UI file and connect them to the handlers defined in a users supplied class.

### Methods

#### gtk\_builder\_new

```
method gtk_builder_new ( --> N-GObject )
```

Creates a new builder object

## **[gtk\_builder\_] new\_from\_file**

```
method gtk_builder_new_from_file ( Str $glade-ui-file --> N-GObject )
```

Creates a new builder object and loads the gui design into the builder

## **[gtk\_builder\_] new\_from\_string**

```
method gtk_builder_new_from_string (
  Str $glade-ui-text, uint32 $length
  --> N-GObject
)
```

Creates a new builder object and takes the gui design from the text argument

## **[gtk\_builder\_] add\_from\_file**

```
method gtk_builder_add_from_file ( Str $glade-ui-file --> int32 )
```

Add another gui design from a file. The result 0 or 1 is returned. 1 means ok.

## **[gtk\_builder\_] add\_from\_string**

```
method gtk_builder_add_from_string (
  Str $glade-ui-text, uint32 $length
  --> int32
)
```

Add another gui design from the text argument. The result 0 or 1 is returned. 1 means ok.

## **[gtk\_builder\_] get\_object**

```
method gtk_builder_get_object ( Str $object-id --> N-GObject )
```

Returns a native widget searched for by its id. See also [GOBJECT:build-id](#).

## **[gtk\_builder\_] get-type-from-name**

```
method gtk_builder_get_type_from_name ( Str $type-name --> int32 )
```

Looks up a type by name. In below example it is shown that this is also accomplished using [GType](#). Furthermore, the codes are not constants! Every new run produces a different gtype code.

```
my GTK::V3::Gtk::GtkBuilder $builder .= new(:filename<my-ui.glade>);
my Int $gtype = $builder.get-type-from-name('GtkButton');
my GTK::V3::Glib::GType $t .= new;
say $t.g-type-name($gtype);           # GtkButton
say $t.from-name('GtkButton');        # $gtype
say $t.g-type-name($t.g-type-parent($gtype)); # GtkBin

#"Depth = 6: Button, Bin, Container, Widget, GInitiallyUnowned, GObject";
say $t.g-type-depth($gtype);          # 6
```

## new

```
multi submethod BUILD ( Str :$filename )
```

Create builder object and load gui design.

```
multi submethod BUILD ( Str :$string )
```

Same as above but read the design from the string.

```
multi submethod BUILD ( Bool :$empty )
```

Create an empty builder.

Generated using Pod::Render, Pod::To::HTML, Camelia™ (butterfly) is © 2009 by Larry Wall