Gnome::Gtk3::Box

A container box

Table of Contents

Description

The `Gnome::Gtk3::Box` widget organizes child widgets into a rectangular area.

The rectangular area of a `Gnome::Gtk3::Box` is organized into either a single row or a single column of child widgets depending upon the orientation. Thus, all children of a `Gnome::Gtk3::Box` are allocated one dimension in common, which is the height of a row, or the width of a column.

`Gnome::Gtk3::Box` uses a notion of packing. Packing refers to adding widgets with reference to a particular position in a `Gnome::Gtk3::Container`. For a `Gnome::Gtk3::Box`, there are two reference positions: the start and the end of the box. For a vertical `Gnome::Gtk3::Box`, the start is defined as the top of the box and the end is defined as the bottom. For a horizontal `Gnome::Gtk3::Box` the start is defined as the left side and the end is defined as the right side.

Use repeated calls to gtk_box_pack_start() to pack widgets into a `Gnome::Gtk3::Box` from start to end. Use gtk_box_pack_end() to add widgets from end to start. You may intersperse these calls and add widgets from both ends of the same `Gnome::Gtk3::Box`.

Because `Gnome::Gtk3::Box` is a `Gnome::Gtk3::Container`, you may also use gtk_container_add() to insert widgets into the box, and they will be packed with the default values for expand and fill child properties. Use gtk_container_remove() to remove widgets from the `Gnome::Gtk3::Box`.

Use gtk_box_set_homogeneous() to specify whether or not all children of the `Gnome::Gtk3::Box` are forced to get the same amount of space.

Use gtk_box_set_spacing() to determine how much space will be minimally placed between all children in the `Gnome::Gtk3::Box`. Note that spacing is added between the children, while padding added by gtk_box_pack_start() or gtk_box_pack_end() is added on either side of the widget it belongs to.

Use gtk_box_reorder_child() to move a `Gnome::Gtk3::Box` child to a different place in the box.

Use gtk_box_set_child_packing() to reset the expand, fill and padding child properties. Use gtk_box_query_child_packing() to query these fields.

Note that a single-row or single-column `Gnome::Gtk3::Grid` provides exactly the same functionality as `Gnome::Gtk3::Box`.

Css Nodes

`Gnome::Gtk3::Box` uses a single CSS node with name box.

In horizontal orientation, the nodes of the children are always arranged from left to right. So `first-child` will always select the leftmost child, regardless of text direction.

See Also

`Gnome::Gtk3::Frame`, `Gnome::Gtk3::Grid`, `Gnome::Gtk3::Layout`

Synopsis

Declaration

```
unit class Gnome::Gtk3::Box;
also is Gnome::Gtk3::Container;
```

Example

Methods

new

```
multi method new ( Bool :$empty! )
```

Create a new empty box.

```
multi method new ( Gnome::GObject::Object :$widget! )
```

Create an object using a native object from elsewhere. See also `Gnome::GObject::Object`.

```
multi method new ( Str :$build-id! )
```

Create an object using a native object from a builder. See also `Gnome::GObject::Object`.

gtk_box_new

Creates a new `Gnome::Gtk3::Box`.

```
method gtk_box_new (
  GtkOrientation $orientation, Int $spacing
  --> N-GObject
)
```

- GtkOrientation $orientation; the box’s orientation.
- Int $spacing; the number of pixels to place by default between children.

Returns N-GObject;

[gtk_box_] pack_start

Adds *child* to *box*, packed with reference to the start of *box*. The *child* is packed after any other child packed with reference to the start of *box*.

```
method gtk_box_pack_start (
  N-GObject $child, Int $expand, Int $fill, UInt $padding
)
```

- N-GObject $child; the `Gnome::Gtk3::Widget` to be added to *box*

- Int $expand; 1 if the new child is to be given extra space allocated to *box*. The extra space will be divided evenly between all children that use this option

- Int $fill; 1 if space given to *child* by the *expand* option is actually allocated to *child*, rather than just padding it. This parameter has no effect if *expand* is set to 0. A child is always allocated the full height of a horizontal `Gnome::Gtk3::Box` and the full width of a vertical `Gnome::Gtk3::Box`. This option affects the other dimension

- UInt $padding; extra space in pixels to put between this child and its neighbors, over and above the global amount specified by reference ends of *box*, then *padding* pixels are also put between

[gtk_box_] pack_end

Adds *child* to *box*, packed with reference to the end of *box*. The *child* is packed after (away from end of) any other child packed with reference to the end of *box*.

```
method gtk_box_pack_end (
  N-GObject $child, Int $expand, Int $fill, UInt $padding
)
```

- N-GObject $child; the `Gnome::Gtk3::Widget` to be added to *box*

- Int $expand; 1 if the new child is to be given extra space allocated to *box*. The extra space will be divided evenly between all children of *box* that use this option

- Int $fill; 1 if space given to *child* by the *expand* option is actually allocated to *child*, rather than just padding it. This parameter has no effect if *expand* is set to 0. A child is always allocated the full height of a horizontal `Gnome::Gtk3::Box` and the full width of a vertical `Gnome::Gtk3::Box`. This option affects the other dimension

- UInt $padding; extra space in pixels to put between this child and its neighbors, over and above the global amount specified by reference ends of *box*, then *padding* pixels are also put between

[gtk_box_] set_homogeneous

Sets the `Gnome::Gtk3::Box`:homogeneous property of *box*, controlling whether or not all children of *box* are given equal space in the box.

```
method gtk_box_set_homogeneous ( Int $homogeneous )
```

- Int $homogeneous; a boolean value, 1 to create equal allotments,

[gtk_box_] get_homogeneous

Returns whether the box is homogeneous (all children are the same size). See gtk_box_set_homogeneous().

```
method gtk_box_get_homogeneous ( --> Int  )
```

Returns Int;

[gtk_box_] set_spacing

Sets the `Gnome::Gtk3::Box`:spacing property of *box*, which is the number of pixels to place between children of *box*.

```
method gtk_box_set_spacing ( Int $spacing )
```

- Int $spacing; the number of pixels to put between children

[gtk_box_] get_spacing

Gets the value set by gtk_box_set_spacing().

```
method gtk_box_get_spacing ( --> Int  )
```

Returns Int;

[gtk_box_] set_baseline_position

Sets the baseline position of a box. This affects only horizontal boxes with at least one baseline aligned child. If there is more vertical space available than requested, and the baseline is not allocated by the parent then extra space available.

```
method gtk_box_set_baseline_position ( GtkBaselinePosition $position )
```

- GtkBaselinePosition $position; a `Gnome::Gtk3::BaselinePosition`

[gtk_box_] get_baseline_position

Gets the value set by gtk_box_set_baseline_position().

```
method gtk_box_get_baseline_position ( --> GtkBaselinePosition  )
```

Returns GtkBaselinePosition;

[gtk_box_] reorder_child

Moves *child* to a new *position* in the list of *box* children. The list contains widgets packed `GTK_PACK_START` as well as widgets packed `GTK_PACK_END`, in the order that these widgets were added to *box*.

```
method gtk_box_reorder_child ( N-GObject $child, Int $position )
```

- N-GObject $child; the `Gnome::Gtk3::Widget` to move
- Int $position; the new position for *child* in the list of children of *box*, starting from 0. If negative, indicates the end of the list

[gtk_box_] query_child_packing

Obtains information about how *child* is packed into *box*.

```
method gtk_box_query_child_packing (
  N-GObject $child, Int $expand, Int $fill,
  UInt $padding, GtkPackType $pack_type
)
```

- N-GObject $child; the `Gnome::Gtk3::Widget` of the child to query
- Int $expand; (out): pointer to return location for expand child property
- Int $fill; (out): pointer to return location for fill child property
- UInt $padding; (out): pointer to return location for padding child property
- GtkPackType $pack_type; (out): pointer to return location for pack-type child property

[gtk_box_] set_child_packing

Sets the way *child* is packed into *box*.

```
method gtk_box_set_child_packing (
  N-GObject $child, Int $expand, Int $fill, UInt $padding,
  GtkPackType $pack_type
)
```

- N-GObject $child; the `Gnome::Gtk3::Widget` of the child to set
- Int $expand; the new value of the expand child property
- Int $fill; the new value of the fill child property
- UInt $padding; the new value of the padding child property
- GtkPackType $pack_type; the new value of the pack-type child property

[gtk_box_] set_center_widget

Sets a center widget; that is a child widget that will be centered with respect to the full width of the box, even if the children at either side take up different amounts of space.

```
method gtk_box_set_center_widget ( N-GObject $widget )
```

- N-GObject $widget; (allow-none): the widget to center

[gtk_box_] get_center_widget

Retrieves the center widget of the box.

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

Returns N-GObject;

Properties

An example of using a string type property of a `Gnome::Gtk3::Label` object. This is just showing how to set/read a property, not that it is the best way to do it. This is because a) The class initialization often provides some options to set some of the properties and b) the classes provide many methods to modify just those properties.

```
my Gnome::Gtk3::Label $label .= new(:empty);
my Gnome::GObject::Value $gv .= new(:init(G_TYPE_STRING));
$label.g-object-get-property( 'label', $gv);
$gv.g-value-set-string('my text label');
```

expand

Whether the child should receive extra space when the parent grows.

Note that the default value for this property is 0 for `Gnome::Gtk3::Box`, but `Gnome::Gtk3::HBox`, `Gnome::Gtk3::VBox` and other subclasses use the old default of 1.

Note that the `Gnome::Gtk3::Widget`:halign, `Gnome::Gtk3::Widget`:valign, `Gnome::Gtk3::Widget`:hexpand and `Gnome::Gtk3::Widget`:vexpand properties are the preferred way to influence child size allocation in containers.

In contrast to `Gnome::Gtk3::Widget`:hexpand, the expand child property does not cause the box to expand itself.

fill

Whether the child should receive extra space when the parent grows.

Note that the `Gnome::Gtk3::Widget`:halign, `Gnome::Gtk3::Widget`:valign, `Gnome::Gtk3::Widget`:hexpand and `Gnome::Gtk3::Widget`:vexpand properties are the preferred way to influence child size allocation in containers.