Skip to content

Panel(bpy_struct)

Basic Panel Example

This script is a simple panel which will draw into the object properties section.

Notice the 'CATEGORY_PT_name' `Panel.bl_idname` , this is a naming convention for panels.

Note

Panel subclasses must be registered for blender to use them.

```
import bpy


class HelloWorldPanel(bpy.types.Panel):
    bl_idname = "OBJECT_PT_hello_world"
    bl_label = "Hello World"
    bl_space_type = 'PROPERTIES'
    bl_region_type = 'WINDOW'
    bl_context = "object"

    def draw(self, context):
        self.layout.label(text="Hello World")


bpy.utils.register_class(HelloWorldPanel)
```

Simple Object Panel

This panel has a `Panel.poll` and `Panel.draw_header` function, even though the contents is basic this closely resembles blenders panels.

```
import bpy


class ObjectSelectPanel(bpy.types.Panel):
    bl_idname = "OBJECT_PT_select"
    bl_label = "Select"
    bl_space_type = 'PROPERTIES'
    bl_region_type = 'WINDOW'
    bl_context = "object"
    bl_options = {'DEFAULT_CLOSED'}

    @classmethod
    def poll(cls, context):
        return (context.object is not None)

    def draw_header(self, context):
        layout = self.layout
        layout.label(text="My Select Panel")

    def draw(self, context):
        layout = self.layout
        box = layout.box()
        box.label(text="Selection Tools")
        box.operator("object.select_all").action = 'TOGGLE'
        row = box.row()
        row.operator("object.select_all").action = 'INVERT'
        row.operator("object.select_random")


bpy.utils.register_class(ObjectSelectPanel)
```

Mix-in Classes

A mix-in parent class can be used to share common properties and `Menu.poll` function.

```
import bpy


class View3DPanel:
    bl_space_type = 'VIEW_3D'
    bl_region_type = 'UI'
    bl_category = "Tool"

    @classmethod
    def poll(cls, context):
        return (context.object is not None)


class PanelOne(View3DPanel, bpy.types.Panel):
    bl_idname = "VIEW3D_PT_test_1"
    bl_label = "Panel One"

    def draw(self, context):
        self.layout.label(text="Small Class")


class PanelTwo(View3DPanel, bpy.types.Panel):
    bl_idname = "VIEW3D_PT_test_2"
    bl_label = "Panel Two"

    def draw(self, context):
        self.layout.label(text="Also Small Class")


bpy.utils.register_class(PanelOne)
bpy.utils.register_class(PanelTwo)
```

base class — `bpy_struct`

class bpy.types.**Panel(bpy_struct)**

Panel containing UI elements

bl_category

The category (tab) in which the panel will be displayed, when applicable

TYPE:

string, default "", (never None)

bl_context

The context in which the panel belongs to. (TODO: explain the possible combinations bl_context/bl_region_type/bl_space_type)

TYPE:

string, default "", (never None)

bl_description

The panel tooltip

TYPE:

string, default ""

bl_idname

If this is set, the panel gets a custom ID, otherwise it takes the name of the class used to define the panel. For example, if the class name is "OBJECT_PT_hello", and bl_idname is not set by the script, then bl_idname = "OBJECT_PT_hello".

TYPE:

string, default "", (never None)

bl_label

The panel label, shows up in the panel header at the right of the triangle used to collapse the panel

TYPE:

string, default "", (never None)

bl_options

Options for this panel type

- `DEFAULT_CLOSED` Default Closed – Defines if the panel has to be open or collapsed at the time of its creation.
- `HIDE_HEADER` Hide Header – If set to False, the panel shows a header, which contains a clickable arrow to collapse the panel and th label (see bl_label).
- `INSTANCED` Instanced Panel – Multiple panels with this type can be used as part of a list depending on data external to the UI. Used tc create panels for the modifiers and other stacks..
- `HEADER_LAYOUT_EXPAND` Expand Header Layout – Allow buttons in the header to stretch and shrink to fill the entire layout width.

TYPE:

enum set in {'DEFAULT_CLOSED', 'HIDE_HEADER', 'INSTANCED', 'HEADER_LAYOUT_EXPAND'}, default {'DEFAULT_CLOSED'}

bl_order

Panels with lower numbers are default ordered before panels with higher numbers

TYPE:

int in [0, inf], default 0

bl_owner_id

The ID owning the data displayed in the panel, if any

TYPE:

string, default "", (never None)

bl_parent_id

If this is set, the panel becomes a sub-panel

TYPE:

string, default “”, (never None)

bl_region_type

The region where the panel is going to be used in

TYPE:

enum in Region Type Items, default ‘WINDOW’

bl_space_type

The space where the panel is going to be used in

TYPE:

enum in Space Type Items, default ‘EMPTY’

bl_translation_context

Specific translation context, only define when the label needs to be disambiguated from others using the exact same label

TYPE:

string, default “*”, (never None)

bl_ui_units_x

When set, defines popup panel width

TYPE:

int in [0, inf], default 0

custom_data

Panel data

TYPE:

`Constraint`, (readonly)

is_popover

TYPE:

boolean, default False, (readonly)

layout

Defines the structure of the panel in the UI

TYPE:

`UILayout`, (readonly)

text

XXX todo

TYPE:

string, default “”, (never None)

use_pin

Show the panel on all tabs

TYPE:

boolean, default False

classmethod poll(context)

If this method returns a non-null output, then the panel can be drawn

RETURN TYPE:

RETURN TYPE:

boolean

draw(context)

Draw UI elements into the panel UI layout

draw_header(context)

Draw UI elements into the panel's header UI layout

draw_header_preset(context)

Draw UI elements for presets in the panel's header

classmethod append(draw_func)

Append a draw function to this menu, takes the same arguments as the menus draw function

classmethod is_extended()

classmethod prepend(draw_func)

Prepend a draw function to this menu, takes the same arguments as the menus draw function

classmethod remove(draw_func)

Remove a draw function that has been added to this menu

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

`bpy.types.Struct` subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- `bpy_struct.id_data`

Inherited Functions

- `bpy_struct.as_pointer`
- `bpy_struct.driver_add`
- `bpy_struct.driver_remove`
- `bpy_struct.get`
- `bpy_struct.id_properties_clear`
- `bpy_struct.id_properties_ensure`
- `bpy_struct.items`
- `bpy_struct.keyframe_delete`
- `bpy_struct.keyframe_insert`
- `bpy_struct.keys`
- `bpy_struct.path_from_id`
- `bpy_struct.path_resolve`

- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values