Skip to content

AddonPreferences(bpy_struct)

```python
bl_info = {
    "name": "Example Add-on Preferences",
    "author": "Your Name Here",
    "version": (1, 0),
    "blender": (2, 65, 0),
    "location": "SpaceBar Search -> Add-on Preferences Example",
    "description": "Example Add-on",
    "warning": "",
    "doc_url": "",
    "tracker_url": "",
    "category": "Object",
}


import bpy
from bpy.types import Operator, AddonPreferences
from bpy.props import StringProperty, IntProperty, BoolProperty


class ExampleAddonPreferences(AddonPreferences):
    # This must match the add-on name, use `__package__`
    # when defining this for add-on extensions or a sub-module of a python package.
    bl_idname = __name__

    filepath: StringProperty(
        name="Example File Path",
        subtype='FILE_PATH',
    )
    number: IntProperty(
        name="Example Number",
        default=4,
    )
    boolean: BoolProperty(
        name="Example Boolean",
        default=False,
    )

    def draw(self, context):
        layout = self.layout
        layout.label(text="This is a preferences view for our add-on")
        layout.prop(self, "filepath")
        layout.prop(self, "number")
        layout.prop(self, "boolean")


class OBJECT_OT_addon_prefs_example(Operator):
    """Display example preferences"""
    bl_idname = "object.addon_prefs_example"
    bl_label = "Add-on Preferences Example"
    bl_options = {'REGISTER', 'UNDO'}
    def execute(self, context):
        preferences = context.preferences
        addon_prefs = preferences.addons[__name__].preferences

        info = "Path: {:s}, Number: {:d}, Boolean {!r}".format(
            addon_prefs.filepath, addon_prefs.number, addon_prefs.boolean,
        )
        self.report({'INFO'}, info)
        print(info)

        return {'FINISHED'}


# Registration
def register():
    bpy.utils.register_class(OBJECT_OT_addon_prefs_example)
    bpy.utils.register_class(ExampleAddonPreferences)


def unregister():
    bpy.utils.unregister_class(OBJECT_OT_addon_prefs_example)
    bpy.utils.unregister_class(ExampleAddonPreferences)
```

base class — `bpy_struct`

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

bl_idname

TYPE:

string, default "", (never None)

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.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.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- Addon.preferences