Skip to content

bpy_extras submodule (bpy_extras.io_utils)

bpy_extras.io_utils.**orientation_helper(axis_forward='Y', axis_up='Z')**

A decorator for import/export classes, generating properties needed by the axis conversion system and IO helpers, with specified default values (axes).

bpy_extras.io_utils.**axis_conversion(from_forward='Y', from_up='Z', to_forward='Y', to_up='Z')**

Each argument us an axis in [‘X’, ‘Y’, ‘Z’, ‘-X’, ‘-Y’, ‘-Z’] where the first 2 are a source and the second 2 are the target.

bpy_extras.io_utils.**axis_conversion_ensure(operator, forward_attr, up_attr)**

Function to ensure an operator has valid axis conversion settings, intended to be used from `bpy.types.Operator.check`.

PARAMETERS:

- **operator** (`bpy.types.Operator`) – the operator to access axis attributes from.
- **forward_attr** (*str*) – attribute storing the forward axis
- **up_attr** (*str*) – attribute storing the up axis

RETURNS:

True if the value was modified.

RETURN TYPE:

bool

bpy_extras.io_utils.**create_derived_objects(depsgraph, objects)**

This function takes a sequence of objects, returning their instances.

PARAMETERS:

- **depsgraph** (`bpy.types.Depsgraph`) – The evaluated depsgraph.
- **objects** (Sequence[`bpy.types.Object`]) – A sequencer of objects.

RETURNS:

A dictionary where each key is an object from `objects`, values are lists of (object, matrix) tuples representing instances.

RETURN TYPE:

dict[`bpy.types.Object`, list[tuple[`bpy.types.Object`, `mathutils.Matrix`]]]

bpy_extras.io_utils.**poll_file_object_drop(context)**

A default implementation for FileHandler poll_drop methods. Allows for both the 3D Viewport and the Outliner (in ViewLayer display mode) to be targets for file drag and drop.

bpy_extras.io_utils.**unpack_list(list_of_tuples)**

bpy_extras.io_utils.**unpack_face_list(list_of_tuples)**

bpy_extras.io_utils.**path_reference(filepath, base_src, base_dst, mode='AUTO', copy_subdir='', copy_set=None, library=None)**

Return a filepath relative to a destination directory, for use with exporters.

PARAMETERS:

- **filepath** (*str*) – the file path to return, supporting blenders relative ‘//’ prefix.
- **base_src** (*str*) – the directory the *filepath* is relative too (normally the blend file).
- **base_dst** (*str*) – the directory the *filepath* will be referenced from (normally the export path).
- **mode** (*str*) – the method used get the path in [‘AUTO’, ‘ABSOLUTE’, ‘RELATIVE’, ‘MATCH’, ‘STRIP’, ‘COPY’]
- **copy_subdir** (*str*) – the subdirectory of *base_dst* to use when mode=’COPY’.
- **copy_set** (*set[tuple[str, str]]*) – collect from/to pairs when mode=’COPY’, pass to *path_reference_copy* when exporting is done.
- **library** (`bpy.types.Library` | None) – The library this path is relative to.

RETURNS:

the new filepath.

RETURN TYPE:

str

bpy_extras.io_utils.**path_reference_copy(copy_set, report=<built-in function print>)**

Execute copying files of path_reference

PARAMETERS:

- **copy_set** (*set[tuple[str, str]]*) – set of (from, to) pairs to copy.
- **report** (*Callable[[str], None]*) – function used for reporting warnings, takes a string argument.

bpy_extras.io_utils.**unique_name(key, name, name_dict, name_max=-1, clean_func=None, sep='.')**

Helper function for storing unique names which may have special characters stripped and restricted to a maximum length.

PARAMETERS:

- **key** (*Any*) – Unique item this name belongs to, name_dict[key] will be reused when available. This can be the object, mesh, material, etc instance itself. Any hashable object associated with the *name*.
- **name** (*str*) – The name used to create a unique value in *name_dict*.
- **name_dict** (*dict*) – This is used to cache namespace to ensure no collisions occur, this should be an empty dict initially and only modified by this function.
- **clean_func** (*function*) – Function to call on *name* before creating a unique value.
- **sep** (*str*) – Separator to use when between the name and a number when a duplicate name is found.

class bpy_extras.io_utils.**ExportHelper**

check(_context)

invoke(context, _event)

class bpy_extras.io_utils.**ImportHelper**

check(_context)

invoke(context, _event)

invoke_popup(context, confirm_text='')