Skip to content

Application Handlers (bpy.app.handlers)

This module contains callback lists

Basic Handler Example

This script shows the most simple example of adding a handler.

```
import bpy


def my_handler(scene):
    print("Frame Change", scene.frame_current)


bpy.app.handlers.frame_change_pre.append(my_handler)
```

Persistent Handler Example

By default handlers are freed when loading new files, in some cases you may want the handler stay running across multiple files (when the handler is part an add-on for example).

For this the `bpy.app.handlers.persistent` decorator needs to be used.

```
import bpy
from bpy.app.handlers import persistent


@persistent
def load_handler(dummy):
    print("Load Handler:", bpy.data.filepath)


bpy.app.handlers.load_post.append(load_handler)
```

Note on Altering Data

Altering data from handlers should be done carefully. While rendering the `frame_change_pre` and `frame_change_post` handlers are called from one thread and the viewport updates from a different thread. If the handler changes data that is accessed by the viewport, this can cause a crash of Blender. In such cases, lock the interface (Render → Lock Interface or `bpy.types.RenderSettings.use_lock_interface` before starting a render.

Below is an example of a mesh that is altered from a handler:

```
def frame_change_pre(scene):
    # A triangle that shifts in the z direction
    zshift = scene.frame_current * 0.1
    vertices = [(-1, -1, zshift), (1, -1, zshift), (0, 1, zshift)]
    triangles = [(0, 1, 2)]

    object = bpy.data.objects["The Object"]
    object.data.clear_geometry()
    object.data.from_pydata(vertices, [], triangles)
```

bpy.app.handlers.**animation_playback_post**

on ending animation playback

bpy.app.handlers.**animation_playback_pre**

on starting animation playback

bpy.app.handlers.**annotation_post**

on drawing an annotation (after)

bpy.app.handlers.**annotation_pre**

on drawing an annotation (before)

bpy.app.handlers.**blend_import_post**

on linking or appending data (after), get a single *BlendImportContext* parameter

bpy.app.handlers.**blend_import_pre**

on linking or appending data (before), get a single *BlendImportContext* parameter

bpy.app.handlers.**composite_cancel**

on a compositing background job (cancel)

bpy.app.handlers.**composite_post**

on a compositing background job (after)

bpy.app.handlers.**composite_pre**

on a compositing background job (before)

bpy.app.handlers.**depsgraph_update_post**

on depsgraph update (post)

bpy.app.handlers.**depsgraph_update_pre**

on depsgraph update (pre)

bpy.app.handlers.**frame_change_post**

Called after frame change for playback and rendering, after the data has been evaluated for the new frame.

bpy.app.handlers.**frame_change_pre**

Called after frame change for playback and rendering, before any data is evaluated for the new frame. This makes it possible to change data and relations (for example swap an object to another mesh) for the new frame. Note that this handler is **not** to be used as 'before the frame changes' event. The dependency graph is not available in this handler, as data and relations may have been altered and the dependency graph has not yet bee updated for that.

bpy.app.handlers.**load_factory_preferences_post**

on loading factory preferences (after)

bpy.app.handlers.**load_factory_startup_post**

on loading factory startup (after)

bpy.app.handlers.**load_post**

on loading a new blend file (after). Accepts one argument: the file being loaded, an empty string for the startup-file.

bpy.app.handlers.**load_post_fail**

on failure to load a new blend file (after). Accepts one argument: the file being loaded, an empty string for the startup-file.

bpy.app.handlers.**load_pre**

bpy.app.handlers.**load_pre**

on loading a new blend file (before).Accepts one argument: the file being loaded, an empty string for the startup-file.

bpy.app.handlers.**object_bake_cancel**

on canceling a bake job; will be called in the main thread

bpy.app.handlers.**object_bake_complete**

on completing a bake job; will be called in the main thread

bpy.app.handlers.**object_bake_pre**

before starting a bake job

bpy.app.handlers.**redo_post**

on loading a redo step (after)

bpy.app.handlers.**redo_pre**

on loading a redo step (before)

bpy.app.handlers.**render_cancel**

on canceling a render job

bpy.app.handlers.**render_complete**

on completion of render job

bpy.app.handlers.**render_init**

on initialization of a render job

bpy.app.handlers.**render_post**

on render (after)

bpy.app.handlers.**render_pre**

on render (before)

bpy.app.handlers.**render_stats**

on printing render statistics. Accepts one argument: the render stats (render/saving time plus in background mode frame/used [peak] memory).

bpy.app.handlers.**render_write**

on writing a render frame (directly after the frame is written)

bpy.app.handlers.**save_post**

on saving a blend file (after). Accepts one argument: the file being saved, an empty string for the startup-file.

bpy.app.handlers.**save_post_fail**

on failure to save a blend file (after). Accepts one argument: the file being saved, an empty string for the startup-file.

bpy.app.handlers.**save_pre**

on saving a blend file (before). Accepts one argument: the file being saved, an empty string for the startup-file.

bpy.app.handlers.**translation_update_post**

on translation settings update

bpy.app.handlers.**undo_post**

on loading an undo step (after)

bpy.app.handlers.**undo_pre**

on loading an undo step (before)

bpy.app.handlers.**version_update**

on ending the versioning code

bpy.app.handlers.**xr_session_start_pre**

on starting an xr session (before)

bpy.app.handlers.**persistent**

Function decorator for callback functions not to be removed when loading new files