

# But how do I stop the server?

- The server will be stopped by:
  - Calling `GenServer.stop()` in the interface section.
  - A callback returning `{:stop, how, state}` (call or cast function).
  - A callback raising an exception.
  - A callback returning an invalid value.
- The function `terminate()` will be called (if present), allowing any shutdown actions to be performed.
- The server name will be de-registered.

```

1 defmodule NameServer6 do
2   @moduledoc """
3     Two ways to stop the server
4     """
5   use GenServer

```

```

7 > def start(name) do=
10 end
11
12 > def add(serverName, name, place) do=
14 end
15
16 > def find(serverName, name) do=
18 end
19

```

```

20 def shutdown(serverName) do
21   GenServer.cast(serverName, :shutdown)
22 end
23
24 def shutdown1(serverName) do
25   GenServer.stop(serverName, :normal)
26 end
27

```

## Interface

```

28 @impl true
29 > def init(_) do=
31 end
32
33 @impl true
34 def terminate(_reason, _state) do
35   # Cleanup code
36 end
37
38 @impl true
39 > def handle_cast({:add, name, place}, state) do=
42 end
43
44 def handle_cast(:shutdown, state) do
45   # do something
46   {:stop, :normal, state}
47 end
48
49 @impl true
50 > def handle_call({:find, name}, _from, state) do=
52 end
53 end

```

## Implementation

Called when the server is terminating because of:

- Callback returns `{:stop, reason, ...}`
- Callback raises an exception.
- Callback returns an invalid value.
- `GenServer.stop(...)` is called.
- ...