```elixir
defmodule NameServer1 do
  @moduledoc """
  Very simple name server supporting two methods:
  add: Add a name and a place.
  find: Given a name, return the place or nil
  """

  def start(name) do
    Process.register(spawn(NameServer1, :loop, [name, %{}]), name)
    :ok
  end

  def rpc(name, request) do
    send(name, {self(), request})

    receive do
      {_name, response} -> response
    end
  end

  def loop(name, state) do
    receive do
      {from, request} ->
        {response, newState} = handleRequest(request, state)
        send(from, {name, response})
        loop(name, newState)
    end
  end

  defp handleRequest({:add, name, place}, state) do
    newState = Map.put(state, name, place)
    {:ok, newState}
  end

  defp handleRequest({:find, name}, state) do
    {state[name], state}
  end
end
```

Constructor.
Spawn a new process, running loop() and register its PID with a name.

Method interface.
Send a message to the server, requesting it to perform an action.

The server.
Preserves the server data (State), and calls the methods and returns the result.

Methods.
The functions that implement the server actions.

NameServer1 Test

- Create the server
- Add a name and place
- Retrieve the place by name

```elixir
defmodule GenServerSlidesTest do
  use ExUnit.Case
  doctest NameServer1

  test "1 - start the server" do
    assert NameServer1.start(:my_server) == :ok
    assert NameServer1.rpc(:my_server, {:add, :dwayne, "Red Dwarf"}) == :ok
    assert NameServer1.rpc(:my_server, {:find, :dwayne}) == "Red Dwarf"
  end
end
```