Be a better parent to your children

Whoami

Herminio Torres

- email: herminiocesar@gmail.com
- github: @herminiotorres
- twitter: @herminiotorres


Setting
Expectations

Things we need to know...

Process

GenServer

Supervisor

ETS(Erlang Term Storage)

Let's go and find out...

PID

```elixir
defmodule Secret.Server do
  # behaviour
  use GenServer
  ...
end
```

```elixir
def start_link(agent_name) do
  GenServer.start_link(__MODULE__, agent_name)
end
```

```elixir
def start_link(agent_name) do
  # agent_name "James Bond"
  GenServer.start_link(__MODULE__, agent_name)
end
```

```
1 {:ok, pid} = Secret.Server.start_link(...)
2 {:ok, #PID<0.189.0>}
```

PID?

- Hard to remember the process
- It time the process crash or stop when start I need to save the new PID

__MODULE__

```elixir
GenServer.start_link(
  __MODULE__,
  agent_name,
  name: __MODULE__ # == Secret.Server
)
```

```
1 {:ok, _} = Secret.Server.start_link(...)
2 {:ok, #PID<0.189.0>}
```

```
1 {:ok, _} = Secret.Server.start_link(...)
2 {:error, {:already_started, #PID<0.189.0>}}
```

```
1 GenServer.whereis(Secret.Server)
2 #PID<0.189.0>
```

__MODULE__?

- Easy to remember the process
- Starting one process

:atom

```elixir
GenServer.start_link(
  __MODULE__,
  agent_name,
  name: naming_process_name(agent_name)
)
```

```elixir
# Ex: input: "Mr. White | output: :mr_white"
def naming_process_name(agent_name) do
  Regex.scan(~r/[^[:punct:]]/, agent_name)
  |> Enum.join
  |> String.trim
  |> String.downcase
  |> String.split
  |> Enum.join("_")
  |> String.to_atom
end
```

```
{:ok, _} = Secret.Server.start_link(
  "Mr. White"
)
{:ok, #PID<0.189.0>}
```

```
{:ok, _} = Secret.Server.start_link(
  "Mr. White"
)
{:error, {:already_started, #PID<0.189.0>}}
```

```elixir
GenServer.whereis(:mr_white)
#PID<0.189.0>
```

```
{:ok, _} = Secret.Server.start_link(
  "James Bond"
)
{:ok, #PID<0.190.0>}
```

:atom?

- Easy to remember the process
- Starting many processes
- The BEAM has a maximum size to handling atoms - 1048576

Registry Module

Registry Concepts:

- A local, decentralized and scalable key-value process storage.
- It allows developers to lookup one or more processes with a given key.
- Each entry in the registry is associated to the process that has registered the key.
- If the process crashes, the keys associated to that process are automatically removed.

Where we use?

Using in :via tuple

it can be used to register and access named processes.

Using as a dispatcher

it can be used to dispatch based on named to triggered from the caller.

Using as a PubSub

it can be used to implement a local, non-distributed, scalable PubSub

Registrations

Looking up, dispatching and registering are efficient and immediate at the cost of delayed unsubscription.

ETS(Erlang Term Storage)

Note that the registry uses one ETS table plus two ETS tables per partition.

How we use?

```elixir
defmodule Secret.Application do
  use Application

  def start(_type, _args) do
    children = [
      ...
      {Registry, [keys: :unique, name: Secret.Registry]}
    ]
    opts = [strategy: :one_for_one, name: Secret.Supervisor]
    Supervisor.start_link(children, opts)
  end
end
```

```elixir
defmodule Secret.Server do
  ...
  def start_link(...)
    ...
    GenServer.start_link(
      __MODULE__,
      agent_name,
      name: via_tuple(agent_name)
    )
  end
  ...
end
```

```elixir
defmodule Secret.Server do
  ...
  # Mr. White
  def via_tuple(agent_name) do
    {:via, Registry, {Secret.Registry, agent_name}}
  end
end
```

```elixir
defmodule Secret.Server do
  ...
  # Server.echo("James Bond", "My name is Bond, James Bond.")
  # "James Bond says: \"My name is Bond, James Bond.\""
  def echo(agent_name, text) do
    GenServer.call(via_tuple(agent_name), {:echo, text})
  end
  ...
end
```

```
{:ok, _} = Secret.Server.start_link(
  "Mr. White"
)
{:ok, #PID<0.189.0>}
```

```elixir
{:ok, _} = Secret.Server.start_link(
  "James Bond"
)
{:ok, #PID<0.190.0>}
```

Supervisor Tree

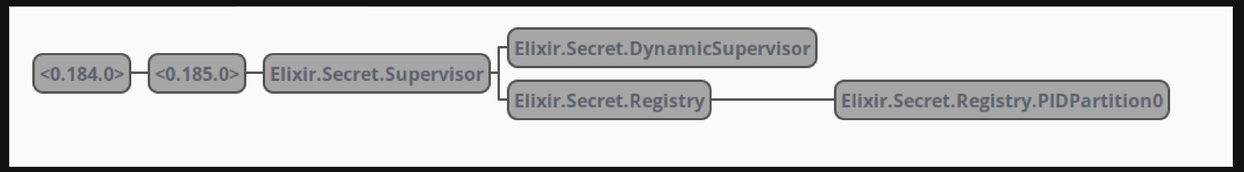
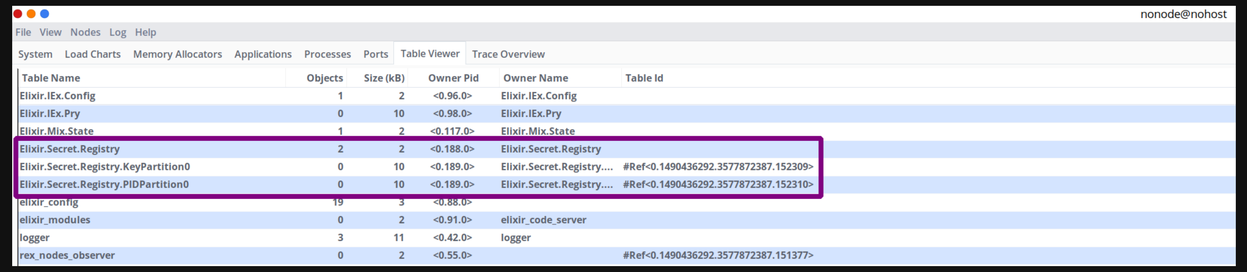
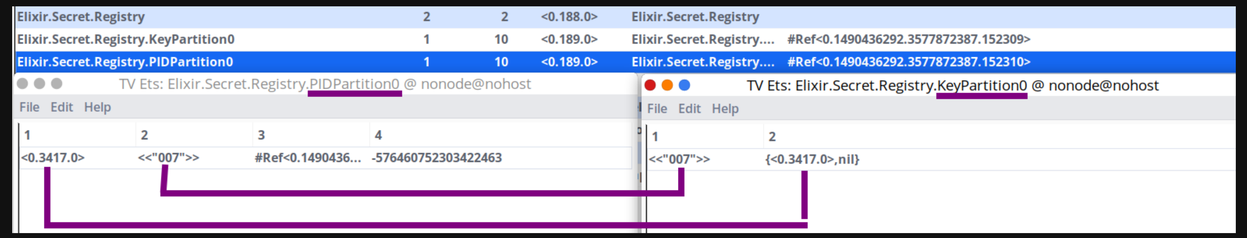


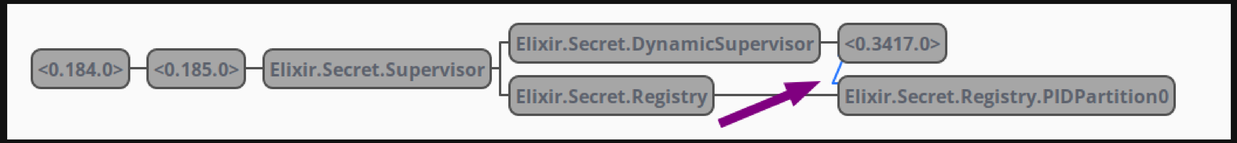
Table Viewer



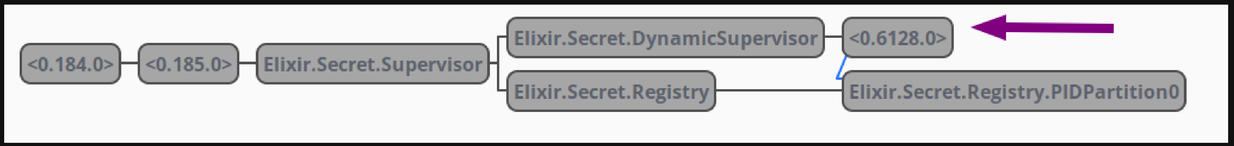
PID Partition and Key Partition



Supervisor Tree - Initialize Process



Supervisor Tree - After Kill Process



```elixir
GenServer.whereis(
  Secret.Server.via_tuple("Mr. White")
)
#PID<0.189.0>
```

PID Partition and Key Partition

1	2
<<"James Bond...	{<0.172.0>,nil}
<<"Mr. White">>	{<0.170.0>,nil}

1	2	3	4
<0.170.0>	<<"Mr. White">>	#Ref<0.1957920...	-576460752303423487
<0.172.0>	<<"James Bond...	#Ref<0.1957920...	-576460752303423423

```
Registry.keys(
  Secret.Registry,
  GenServer.whereis(
    Secret.Server.via_tuple("James Bond")
  )
)
["James Bond"]
```

```
Registry.lookup(
  Secret.Registry,
  "James Bond"
)
[{#PID<0.189.0>, nil}]
```

Thank you!

https://unsplash.com/photos/kzQ6gbTR-Fg

References:

- Registry docs: https://hexdocs.pm/elixir/master/Registry.html
- Elixir — Process Registries: https://medium.com/@StevenLeiva1/elixir-process-registries-a27f813d94e3

References:

- Multiplayer Go with Elixir's Registry, PubSub and dynamic supervisors: https://blog.appsignal.com/2019/08/13/elixir-alchemy-multiplayer-go-with-registry-pubsub-and-dynamic-supervisors.html

- How to start processes with dynamic names in Elixir: https://thoughtbot.com/blog/how-to-start-processes-with-dynamic-names-in-elixir

References:

- Using the Registry in Elixir 1.4: https://medium.com/elixirlabs/registry-in-elixir-1-4-0-d6750fb5aeb

- Process registry in Elixir: a practical example: https://www.brianstorti.com/process-registry-in-elixir/

References:

- Registry in Elixir 1.4 - Youtube Video: https://www.youtube.com/watch?v=VG8bBnOYj2g

- Elixircasts - 109 - DynamicSupervisor and Registry: https://elixircasts.io/dynamicsupervisor-and-registry

References:

- Process pools with Elixir's Registry: https://andrealeopardi.com/posts/process-pools-with-elixirs-registry/

- Darth Vader Image - Unsplash: https://unsplash.com/photos/GV2JHwil21U

References:

- The Batman Image - Wikipedia: https://en.wikipedia.org/wiki/Batman_Begins

- The Machinist Image - Wikipedia: https://en.wikipedia.org/wiki/The_Machinist

Thank you!

Herminio Torres

- email: herminiocesar@gmail.com
- github: @herminiotorres
- twitter: @herminiotorres