Process Name Registration

Julian Doherty
@madlep

“Process name registration”

- sounds super boring

- TL;DR - DNS for BEAM (Elixir) Processes

let it crash

```
my_pid = Foo.start_link
Foo.bar(my_pid, “a message”)
Foo.bar(my_pid, “another message”)

# OH NO! CRASH!

Foo.bar(my_pid, “hello, world!”)
# nothing happens…
# SAD!
```

So we register

```
my_pid = Foo.start_link
Process.register(my_pid, :some_name)
Foo.bar(:some_name, “a message”)
Foo.bar(:some_name, “another message”)

# OH NO! CRASH!
# manually handle it…

my_pid2 = Foo.start_link
Process.register(my_pid2, :some_name)

send(:some_name, “hello, world!”)
# huge
```

This is what supervisors do

```
my_sup = Supervisor.start_link(
  Supervisor.child_spec(Foo, name: :some_name)
)
Foo.bar(:some_name, “a message”)
Foo.bar(:some_name, “another message”)

# OH NO! CRASH!
# supervisor does the magic

Foo.bar(:some_name, “hello, world!”)
# bigly
```

Local name registration

- Default is to use the built in BEAM name registration features
- Local VM
- easy to use
- robust

Local name registration

- “global” - as in “global variables”
  (not as in distributed systems)
- as in “global variables are bad”
- makes tests harder
- code more brittle
- can’t namespace

Elixir Registry

- https://hexdocs.pm/elixir/Registry.html
- part of Elixir stdlib
- allows creation of multiple namespaces, each independent
- we use this for running multiple parallel event sourcing projectors, each with separate data looked up by id

Elixir Registry

```
{:ok, _} = Registry.start_link(
  keys: :unique, name: :some_registry
)

# my_sup = Supervisor.start_link(
#   Supervisor.child_spec(Foo, name: :some_name)
# )

my_sup = Supervisor.start_link(
  Supervisor.child_spec(Foo, name:
    {:via, Registry, {:some_registry, :some_name)
)
```

Global

- http://erlang.org/doc/man/global.html
- part of Erlang stdlib
- same semantics as local, but distributed
- simple
- not flexible enough for a lot of use cases
- great for built in Erlang stuff
- can be slow

Global

```
my_sup = Supervisor.start_link(
  Supervisor.child_spec(Foo, name: {:global,:some_name})
)
Foo.bar({:global,:some_name}, “a message”)
Foo.bar({:global,:some_name}, “another message”)

# OH NO! CRASH!
# supervisor does the magic

Foo.bar({:global,:some_name}, “hello, world!”)
# bigly
```

gproc

- local with more stuff
- very mature
- can do global, but gets weird

syn

- designed for lots of processes, lots of join/leave
- IOT devices is use case
- assumes relatively static cluster

swarm

- lots of cool cloud stuff
- designed for lots of processes, lots of join/leave
- can automatically distribute/rebalance processes
- may be overkill