

Deumbrellization ✨

Artur Sulej



Artur Sulej @ Fresha.com



github.com/Artur-Sulej



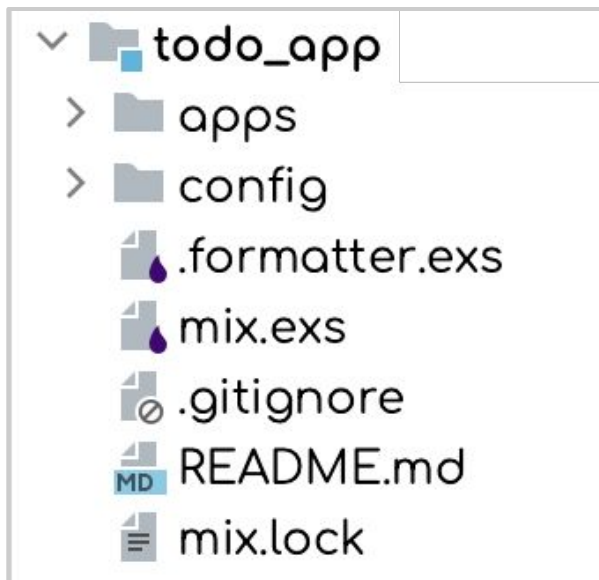
linkedin.com/in/artursulej

Why?

- Independent applications
- Faster compile time
- Scalability
- Local development
- Less effort than e.g. gRPC

Umbrella

- Subapps
- One config
- Deps



```
# apps/todo_web/mix.exs
defp deps do
  [
    {:todo, in_umbrella: true},
    # ...
  ]
end
```

Splitting


- Remove `in_umbrella` dependencies
- Separate configs
- Adjust `mix.exs`
- Set up communication

Communication

```
defmodule TodoWeb.TaskController do
  use TodoWeb, :controller

  alias Todo.TODOLists

  def index(conn, _params) do
    tasks = TODOLists.list_tasks()
    render(conn, "index.html", tasks: tasks)
  end
end
```



We can't just invoke this code, because it's no longer available locally.

Nodes

- A **node** is an instance of the Erlang Virtual Machine.
- Nodes can communicate between each other.
- Together they form distributed system.

```
todo > iex --sname tomato -S mix
iex(tomato@MacBook-Pro-8)1> Node.list
[]
iex(tomato@MacBook-Pro-8)2> Node.connect(:"cucumber@MacBook-Pro-8")
true
iex(tomato@MacBook-Pro-8)3> Node.list
[: "cucumber@MacBook-Pro-8"]
```

```
todo_web > iex --sname cucumber -S mix
iex(cucumber@MacBook-Pro-8)1> Node.list
[]
iex(cucumber@MacBook-Pro-8)2> Node.list
[: "tomato@MacBook-Pro-8"]
```

libcluster

A library that provides a mechanism for automatically forming clusters of nodes.

```
defmodule TodoWeb.Application do
  @moduledoc false
  use Application

  @impl true
  def start(_type, _args) do
    topologies = [
      local_epmd: [
        strategy: Cluster.Strategy.LocalEpmd
      ]
    ]

    children = [
      {Cluster.Supervisor, [topologies, [name: TodoWeb.ClusterSupervisor]]},
      # ...
    ]

    opts = [strategy: :one_for_one, name: TodoWeb.Supervisor]
    Supervisor.start_link(children, opts)
  end
end
```


:rpc

Erlang has a built-in library `:rpc` for remote calls.

```
iex(todo_web@MacBook-Pro-8)10> :rpc.call("deumbrellization-todo@MacBook-Pro-8",
Todo.ToDoLists, :list_tasks, [])
[debug] QUERY OK source="tasks" db=0.7ms idle=542.5ms
SELECT t0."id", t0."description", t0."inserted_at", t0."updated_at" FROM "tasks"
AS t0 []
↳ :erpc.execute_call/4, at: erpc.erl:589
[
  %{
    __meta__: #Ecto.Schema.Metadata<:loaded, "tasks">,
    __struct__: Todo.ToDoLists.Task,
    description: "Fix tap",
    id: 24,
    inserted_at: ~N[2023-02-21 23:41:20],
    updated_at: ~N[2023-02-21 23:41:20]
  }
]
```

```
defmodule TodoWeb.TaskController do
  use TodoWeb, :controller

  - alias Todo.ToDoLists
  + @node : "todo@MacBook-Pro-3"

  def index(conn, _params) do
    - tasks = TodoLists.list_tasks()
    + tasks = :rpc.call(@node, Todo.ToDoLists, :list_tasks, [])
    +
    render(conn, "index.html", tasks: tasks)
  end
end
```

Syntactic sugar

```
defmodule RemoteCallMacro do
  defmacro remote(node_name, do: ast) do
    # {::., [line: 10], [{:__aliases__, [line: 10], [:Todo, :TodoLists]}, :list_tasks]}, [], []}
    {::., _, [{_, _, aliases}, function_name]}, _, args} = ast
    module = Module.concat(aliases)

    quote do
      :rpc.call(unquote(node_name), unquote(module), unquote(function_name), unquote(args))
    end
  end
end
```

```
defmodule TodoWeb.TaskController do
  use TodoWeb, :controller

  import RemoteCallMacro
  @node Application.compile_env!(:todo_web, [:nodes, :todo, :short_name])

  def index(conn, _params) do
    tasks =
      remote(@node) do
        Todo TodoLists.list_tasks()
      end

    render(conn, "index.html", tasks: tasks)
  end
end
```

```

defmodule RemoteCallMacro do
  defmacro remote(node_short_name, do: ast) do
    {{:., _, [{_, _, aliases}, function_name]}, _, args} = ast
    module = Module.concat(aliases)

    quote do
      node_name =
        Enum.find(Node.list(), fn node →
          node ▷ Atom.to_string() ▷ String.split("@") ▷ hd() = unquote(node_short_name)
        end)

      node_name || raise "Error node not found: #{inspect(unquote(node_short_name))}"
      response = :rpc.call(node_name, unquote(module), unquote(function_name), unquote(args))

      case response do
        {:badrpc, _} →
          raise "Error calling node #{inspect(node_name)} via rpc: #{inspect(response)}"

        response →
          response
      end
    end
  end
end

```

Tests

Strategies:

- Running nodes
- Dummy implementation
- `{:app, in_umbrella: true, only: [:test]}`
- Extend macro and mock

Tests – mock

```
defmodule TodoWeb.TaskControllerTest do
  use TodoWeb.ConnCase

  describe "index" do
    test "lists all tasks", %{conn: conn} do
      Process.put(
        {Todo.TodoLists, :list_tasks, []},
        [
          %{
            __struct__: Todo.TodoLists.Task,
            id: 5,
            description: "Do some stuff"
          }
        ]
      )

      conn = get(conn, Routes.task_path(conn, :index))
      assert html_response(conn, 200) =~ "Do some stuff"
    end
  end
end
```

```
defmodule RemoteCallMacro do
  defmacro remote(node_short_name, do: ast) do
    {{:., _, [{_, _, aliases}, function_name]}, _, args} = ast
    module = Module.concat(aliases)

    if Mix.env() == :test do
      quote do
        Process.delete({unquote(module), unquote(function_name), unquote(args)})
      end
    else
      quote do

```

Why not?

- No isolation
- Elixir/Erlang only
- Not explicit
- Deumbrellization is not despaghettization

Resources

- Anton Mishchuk:
<https://medium.com/matic-insurance/designing-scalable-application-with-elixir-from-umbrella-project-to-distributed-system-42f28c7e62f1>
- Karol Słuszniaak:
<https://heyplay.io/blog/building-multiplayer-gaming-gamedev-platform-with-elixir-phoenix-liveview-and-rust>
- libcluster: <https://github.com/bitwalker/libcluster>
- Repo: <https://github.com/Artur-Sulej/deumbrellization>

