

Beyond request-response cycle

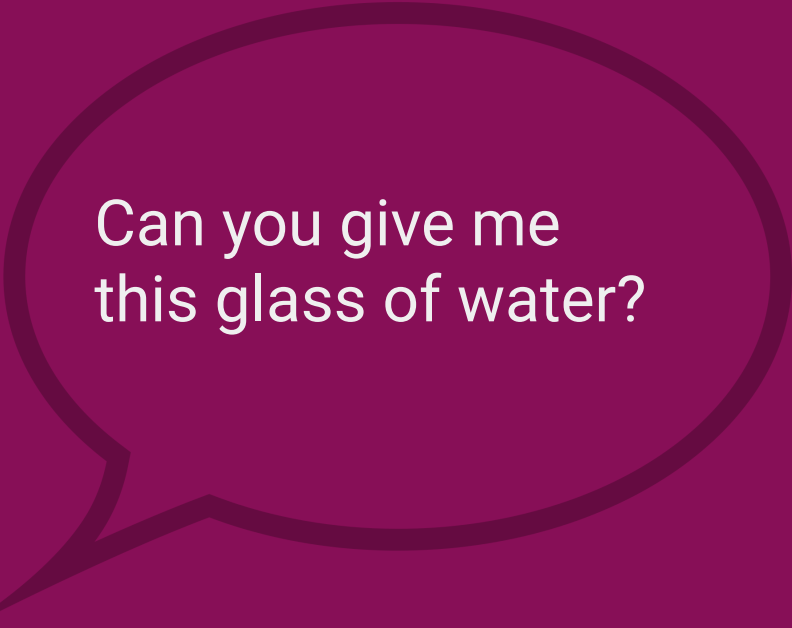
Understanding request-response cycle



How much is 2 times 2?



4!



Can you give me
this glass of water?



Here you are

But there is always that one guy...

$$\int_1^3 \frac{\sin(\sqrt{x} + a)e^{\sqrt{x}}}{\sqrt{x}} dx$$



This site can't be reached

integralsolved.com's server IP address could not be found.

- Did you mean <http://integralsolve.com/>?
- Search Google for [integral solved](#)

ERR_CONNECTION_TIMED_OUT

Problems connected to background processing

Growing codebase

Overcomplicated, stateful code

Extra dependencies

Segregating responsibilities

Solutions?

Part III



```
➔ ~ nohup
```



Sidekiq

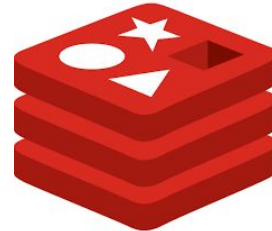
nosql



Crontab



mongoDB



Part III



elixir

How Elixir deals with it?

- OTP
- Simple codebase
- Support in standard library
- No need for extra dependencies

Simplest and easiest solution



```
1  defmodule BGJobs.Statistics.Server do
2    def start_link() do
3      Task.start_link(fn -> loop() end)
4    end
5
6    defp loop() do
7      process()
8      Process.sleep(1000)
9      loop()
10   end
11
12   defp process(), do: Logger.info("processing!")
13 end
```

- Closed for extension
- Hard to customize
- Basically “cron job”
- Deploy or error can break flow

```
1 defmodule BGJobs.Statistics.Server do
2   use GenServer
3
4   def start_link() do
5     GenServer.start_link(__MODULE__, [], name: __MODULE__)
6   end
7
8   def init(_) do
9     {:ok, Process.send_after(self(), :process, 1000)}
10  end
11
12  def handle_info(:process, _) do
13    process()
14    Process.send_after(self(), :process, 1000)
15    {:noreply, nil}
16  end
17
18  defp process(), do: Logger.info("processing!")
19 end
```

- Easily extendable
- Cast, Call, Info separation
- Reliable, reusable and well-known solution



```
1  defmodule BGJobs.Statistics.Server do
2    use GenServer
3
4    def start_link() do
5      GenServer.start_link(__MODULE__, [], name: __MODULE__)
6    end
7
8    def init(_, do: {:ok, nil})
9
10   def handle_cast(:process, _) do
11     process()
12     {:noreply, nil}
13   end
14
15   defp process(), do: Logger.info("processing!")
16 end
```

But beware

Nothing is perfect

```
1 defmodule BGJobs.Statistics.ServerFlood do
2   require Logger
3   use GenServer
4
5   def start_link() do
6     GenServer.start_link(__MODULE__, [], name: __MODULE__)
7   end
8
9   def init(_, do: {:ok, nil})
10
11   def handle_cast(:process, _) do
12     process()
13     self() |> Process.info(:message_queue_len) |> inspect() |> Logger.info()
14     {:noreply, nil}
15   end
16
17   defp process(), do: Enum.map(1..100_000, &:rand.uniform/1)
18
19   def flood() do
20     GenServer.cast(__MODULE__, :process)
21     Process.sleep(10)
22     flood()
23   end
24 end
```



```
iex(1)> BGJobs.Statistics.ServerFlood.flood  
  
17:54:15.693 [info]  {:message_queue_len, 3}  
  
17:54:15.725 [info]  {:message_queue_len, 5}  
  
17:54:15.760 [info]  {:message_queue_len, 8}  
  
17:54:15.791 [info]  {:message_queue_len, 9}  
  
17:54:15.821 [info]  {:message_queue_len, 11}  
  
17:54:15.852 [info]  {:message_queue_len, 13}  
  
17:54:15.883 [info]  {:message_queue_len, 15}  
  
17:54:15.920 [info]  {:message_queue_len, 17}  
  
17:54:15.949 [info]  {:message_queue_len, 19}  
  
17:54:15.981 [info]  {:message_queue_len, 20}
```

Genstage

Summary

Thank you!